all rights reversed

HACKER

Νοέμβρης 2013 # Τιμή τεύχους 4€

τεύχος 026 - BitDreaming Edition

Πηγαίνετε κατευθείαν στη φυλακή [*]



Γνωρίστε και λατρέψτε τη Markdown

Μάθετε πώς θα σερφάρετε *χωρίς* το δικό σας IP

Στήστε ένα φθηνό, εξαιρετικά ικανό bitcoin mining rig

[*] Αυτή που έχουμε υπόψη θα σας αρέσει :P

all rights reversed

HACKER

tefhos 26o

Περιεχόμενα

και οδηγίες χρήσης

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Πέτρος Κυλαδίτης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55133, Καλαμαριά. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/subscriptions και οι ίδιες οι παραγγελίες γίνονται από το http://deltahacker.gr/pdf12order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) Το deltaHacker δεν είναι μόνο ένα μηνιαίο, kick-ass περιοδικό :P
http://deltahacker.gr/category/addenda
http://deltahacker.gr/category/deltacast
http://deltahacker.gr/category/dev-random



Πλήρη άρθρα που συμπληρώνουν το περιοδικό [**addenda**]. Το video podcast [**netcasting → deltaCast**]. Άρθρα που μας ήρθε να γράψουμε, έτσι, απλά [**/dev/random**]. Τα προκλητικά **mind hacks**. Τα βίντεο από τα προηγούμενα live streams [**netcasting → deltaStream**]. Ολόκληρα blasts from the past [**bftp**]. Τα πάντα ενδιαφέροντα **forums**. Το σημείο συνάντησης για τους φίλους του περιοδικού.

deltahacker.gr

Αληθινό, διασκεδαστικό hacking **για όλους**!

Ο καμβάς, πάνω στον οποίο κι εσείς σχεδιάζετε

Διαβάζω καμιά φορά editorials — περισσότερες φορές από το υπόλοιπο του περιοδικού που τα φιλοξενεί, ομολογώ. Συνήθως γιατί το editorial είναι ανεξάρτητο από τον γενικότερο προσανατολισμό ή τη γραμμή, αν θέλετε, του ιδίου του περιοδικού. Βρίσκεις έτσι διαμάντια σε άσχετα περιοδικά, μιας κι ο συντάκτης γράφει σε προσωπικό επίπεδο κι αν μη τι άλλο τον γνωρίζεις καλύτερα.

by @arkoudos

Αυτό το editorial που τώρα διαβάζετε δεν είναι σαν άλλα που διαβάζω, αφού φιλοξενούμαι. Αποτελεί μια θαυμάσια ευκαιρία να μιλήσω σε ανθρώπους σαν εμένα. Αλλά ας το πάρω πίσω αυτό κι ας αναδιατυπώσω: Ανθρώπους σαν εμένα, οι οποίοι αρχίζουν να διαβάζουν το editorial και είναι έτοιμοι να διαβάσουν πράγματα άσχετα με τα θέματα στις σελίδες που ακολουθούν.

Αφού λοιπόν έχω την ευκαιρία να φιλοξενηθώ στο editorial αυτού του τεύχους, ας μιλήσω ως άνθρωπος που απευθύνεται σε ανθρώπους με δύναμη.

Δηλαδή σ' εσάς. Σ' εσένα.

Άκου ρε 'συ, φίλε, να δεις τι γίνεται.

Αυτός ο κόσμος άλλαξε πολύ τα τελευταία χρόνια. Το ξέρω, γιατί όπως κι αρκετοί από εσάς, ήμουν κι εγώ στο μέσο αυτής της αλλαγής. Η γενιά μου δεν είχε κινητό, είχε μόλις δύο κανάλια, δεν είχε Ίντερνετ, δεν είχε πολλές εφημερίδες. Καταλαβαίνω, πρώτος εγώ, ότι η γενιά μου είχε άλλα ενδιαφέροντα πράγματα. Μπορεί να ήταν σημαντικά, μπορεί και όχι. Αλλά δεν είχε όλα αυτά που έχουν νεότερες γενιές.

Έχει σημασία; Έχει — και μάλιστα μεγάλη. Γιατί, αν έχεις μεγαλώσει σε έναν κόσμο όπου επικοινωνείς μόνο με τους διπλανούς σου, που για να μιλήσεις με κάποιον ο οποίος είναι μακριά σου πρέπει και οι δύο να είστε ταυτόχρονα σπίτι σας, που η ενημέρωση έρχεται μόνο από συγκεκριμένες, εγκεκριμένες πηγές, μπορείς να καταλάβεις ότι οι καιροί έχουν αλλάξει δραματικά.

Και άλλαξαν δραματικά και προς τις *δύο* κατευθύνσεις.

Αποκτήσαμε, εμείς, ο λαός, τη δύναμη να ελέγχουμε καλύτερα, να μαθαίνουμε ειδήσεις που παλιότερα δεν μπορούσαμε, να εκφραζόμαστε απείρως πιο ελεύθερα — αλλά και η έκφρασή μας να υπόκειται σε απείρως μεγαλύτερη κριτική. Μπορούμε πλέον να μιλάμε με αγνώστους όπου κι αν βρισκόμαστε κι όπου κι αν βρίσκονται, μπορούμε να αποκτάμε πρόσβαση σε πληροφορία που παλιότερα ήταν σαν να μην υπήρχε, μπορούμε να βάζουμε σε μια σειρά και να οριοθετούμε τις σκέψεις μας έχοντας πρώτα —αν έχουμε τη διάθεση— βουτήξει στη σκέψη κάποιου άλλου, τον οποίο πιθανώς ούτε καν γνωρίζουμε.

Την ίδια στιγμή απέκτησαν, όλοι εκείνοι που ενοχλούνται από τις ελευθερίες και τις δυνατότητες που έχουμε, την ικανότητα ελέγχου τως γραμμών επικοινωνίας μας. Για παράδειγμα, εταιρείες όπως η Microsoft, η Apple και η Google, δίνουν κάθε στοιχείο ηλεκτρονικής μας σκέψης σε όποιον έχει την ισχύ να ζητήσει δεδομένα που μας αφορούν για να τα μελετήσει, ενδεχομένως να τα αλλοιώσει ή/και αύριο να τα χρησιμο-

ποιήσει εναντίον μας. Οι εταιρείες μας παρακολουθούν στο Διαδίκτυο. Την ίδια στιγμή, κάθε κινητό που δεν έχει αποσπώμενη μπαταρία αποτελεί ένα μικρόφωνο κι ένα σημείο αναφοράς στο χάρτη για όλους όσοι έχουν τη δύναμη να το ελέγχουν — αλλά εμείς δεν έχουμε απαραίτητα τη δύναμη να τους ελέγχουμε.

Μολαταύτα, κάτι που δεν οριοθετείται είναι η διάθεση για ελευθερία. Όσοι είστε κοινωνοί της ηλεκτρονικής δημιουργικότητας, είτε "απαλλοτριώνοντας" ένα εργαλείο (βλ., π.χ., rooting ή jailbreak) ώστε να δουλεύει ακριβώς όπως θέλει ο κάτοχός του κι όχι αποκλειστικά όπως ονειρεύεται ο κατασκευαστής του, είτε όσοι ανακατεύεστε με ένα software ώστε να κάνει αυτό που θέλει ο χρήστης του κι όχι αυτό που θέλει ο εμπορικός του εκμεταλλευτής, είτε όσοι βοηθάτε ώστε οι ιδέες να μη μένουν κλειδωμένες στο πλαίσιο παράλογων πνευματικών δικαιωμάτων αλλά να χρησιμοποιούνται για την καλυτέρευση της ζωής όλων, όλοι εσείς είστε άνθρωποι οι οποίοι, εν αγνοία σας ή όχι, ζωγραφίζετε τον καμβά ενός άλλου, διαφορετικού κόσμου.

Το τι θα εμφανιστεί πάνω στον καμβά, είναι δική σας δουλειά. (Σ.τ.Ε. Είναι;) Το σίγουρο είναι ότι με τις γνώσεις σας, καθώς και με τη διάθεσή σας να δημιουργήσετε, μπορείτε είτε να ελευθερώσετε τον κόσμο είτε να τον φυλακίσετε χειρότερα από πριν. Τα παλιά χρόνια, για να φτιάξεις ένα χρήσιμο εργαλείο έπρεπε να "κλέψεις" την τέχνη του σιδηρουργού ή του ξυλουργού. Αυτό ήταν απελπιστικά δύσκολο, αφού η μετάδοση των γνώσεων ήταν τρομερά περιορισμένη και μυστικοπαθής.

Σήμερα, η γνώση της δημιουργίας είναι κοινή. Ένας κοινός χώρος επικοινωνίας, χιλιάδες διαφορετικά μυαλά, εκατομμύρια ιδέες και σκέψεις κάθε φορά, δισεκατομμύρια απορίες και προβλήματα που περιμένουν να λυθούν. Εσείς, που έχετε το επίπεδο της βαθιάς γνώσης της νέας μαγείας που ονομάζεται "τεχνολογία", έχετε στα δάκτυλα των χεριών σας και στους νευρώνες του μυαλού σας τη δυνατότητα να αλλάξετε τον κόσμο για όλους μας.

Σκέψου το, λοιπόν, φίλε. Εσύ, για ολόκληρο τον κόσμο.

Για μένα που όλα αυτά είναι μυστήρια και αδυνατώ να κατανοήσω ακόμη και πώς συνδέεται ένας σκληρός δίσκος στον υπολογιστή, ξέρω μολαταύτα ότι κάποιος, εκεί έξω, θα έχει τη δυνατότητα να επινοήσει ένα δίκτυο που δεν θα ελέγχεται και δεν θα λογοκρίνεται από κανέναν, μέσα από το οποίο θα μεταφέρω τις σκέψεις μου σε ολόκληρο τον κόσμο. Κάποιος άλλος, θα φτιάξει έναν 3D printer που θα μου παρέχει τη δυνατότητα να δημιουργώ ένα παιχνίδι για τις κόρες μου όπως ακριβώς το φαντάστηκα, όμως θα υπάρχει κι εκείνος που θα είναι σε θέση να με φυλακίσει, για ένα προσωπικό μήνυμα που αντάλλαξα με έναν άγνωστο στην άλλη άκρη του κόσμου ή για έναν δικτυακό τόπο που επισκέφτηκα ενώ "δεν έπρεπε".

Πλήρης (ηλεκτρονικός) έλεγχος και πλήρης εξουσία: για όλους — ή για λίγους.

Ο κόσμος γύρω μας είναι πια εν πολλοίς ηλεκτρονικός, κι εσείς που διαρκώς αναρωτιέστε πώς δουλεύει και πώς αλλάζει είστε οι μηχανικοί του. Οι μηχανικοί ενός πολύ αληθινού, αυτή τη στιγμή, παιχνιδιού.

Σε ζηλεύω λοιπόν λίγο — είναι εμφανές. Κι όσο την ίδια στιγμή να σε φοβάμαι και λίγο, χαίρομαι ειλικρινά που αναρωτιέσαι. Ίσως μόνο ελπίζω να έχουμε τις ίδιες απορίες...



σνέιλ μέιλ

Αγαπητοί μου,

Η μέρα σήμερα προχωρά κανονικά, χωρίς πολλές πολλές εκπλήξεις. Από το πρωί είχαμε όλοι μαζευτεί στο τραπέζι, άλλος παίρνοντας το πρωινό του, άλλος διαβάζοντας τα αφιερώματα των εφημερίδων, άλλος καθαρίζοντας και λαδώνοντας την γραφομηχανή του, και πετάγαμε ιδέες για τη συνέχεια.

Κυριακή, δεκαεφτά του μηνού σήμερα, κι εγώ ήθελα να περάσω από Μοναστηράκι να χαζέψω στα παλαιοβιβλιοπωλεία. Η μητέρα της εξαδέλφης συμφωνούσε. Κλαιγόταν που πάνε να της τελειώσουν τα μπαχαρικά και ήθελε, επί τη ευκαιρία, να κατέβει από Ευριπίδου, για κουρκουμά, γκάραμ μασάλα και φύλλα κάρυ.

Η Μαιρούλα ήθελε να πάει για ψώνια -- ή, μάλλον, να προσποιηθεί ότι πηγαίνει για ψώνια, γιατί πού λεφτά για ψώνια την σήμερον ημέρα. Τέλος πάντων, να πάει στα μαγαζιά ήθελε και ήταν στεναχωρημένη που τα μαγαζιά σήμερα είναι κλειστά. «Κυριακή και θέλεις μαγαζιά, ε; Καλόμαθες, κι εσύ, ε;» την πειράζαμε.

Η εξαδέλφη δεν είχε κάποια συγκεκριμένη ιδέα για το πού θα ήθελε να πάει, αλλά ήξερε πολύ καλά πού *δεν* θα ήθελε να πάει. Δεν ήθελε με τίποτα να περάσει από Πατησίων στο ύψος της Στουρνάρα, μάλλον μεταξύ Στουρνάρα και Τοσίτσα, εκεί λίγο πιο πριν από το Αρχαιολογικό Μουσείο.

Στην αρχή δεν καταλάβαμε γιατί. Μέχρι που μας έδειξε τις εφημερίδες.

Ετσι, με τα πολλά, καθώς η ιδέα για κέντρο δεν έπαιζε πλέον, αποφασίσαμε να ακολουθήσουμε την πρόταση της κυρίας Καίτης, και να κατέβουμε στο Φάληρο για μεζέδες, σουλατσάρισμα, και γλυκό. Είχε ακούσει για ένα φτηνό πολίτικο ουζάδικο / μεζεδοπωλείο πολύ κοντά στην Αγίου Αλεξάνδρου και μας έψησε να το δοκιμάσουμε.

Αυτό λοιπόν είναι το σχέδιο για το υπόλοιπο της μέρας. Πρώτα ούζο και μεζέδες, και μετά θα πάμε προς Εντεμ για τούρκικο καφέ, καζάν ντιμπί και ηλιοβασίλεμα.

Προς το παρόν, πάντως, παραμένουμε καθισμένοι στο τραπέζι.

Ετοιμαζόμουν να γράψω ένα σχόλιο στο τελευταίο editorial του κυρίου Αρχισυντάκτη (TM), αλλά με διέκοψε η Μαιρούλα με τον καημό της.

«Μα γιατί ρε παιδιά, γιατί να μην είναι ανοιχτά τα μαγαζιά τις Κυριακές; Πείτε μου. Γιατί διαφωνούνε όλοι; Αλλού γιατί ανοίγουνε τα μαγαζιά;»

«Αλλού είναι αλλού. Εδώ πάντως οι υπάλληλοι φοβούνται ότι θα χάσουνε το ρεπό τους. Και τα μικρομάγαζα φοβούνται πως θα τους καταπιούνε οι μεγάλες αλυσίδες και τα σουπερμάρκετ», σχολίασε η κυρία Καίτη.

«Ε, δίκιο δεν έχουν;» αναρωτήθηκε η μητέρα της εξαδέλφης.

«Κοιτάξτε, δεν ξέρω πόσο δίκιο έχουν», είπα εγώ. «Στην Αγγλία πάντως, που τα μαγαζιά είναι ανοιχτά τις Κυριακές, όλες τις Κυριακές εδώ και χρόνια, ο τόπος είναι γεμάτος και μαγαζιά και μικρομάγαζα.»

«Αμάν πιά με αυτή την Αγγλία. Αγγλία και Αγγλία. Ουφ. Στην Αγγλία έχουνε λεφτά και δεν τους νοιάζει», επέμεινε η μητέρα της εξαδέλφης. «Γι αυτό και δεν κόβουνε τα ρεπό, πληρώνουνε τις υπερωρίες, κάνουνε προσλήψεις. Έχουν λεφτά και μπορούν. Κι εμείς αν είχαμε...»

«Εμείς; Χά! Θα γελάσω τώρα», είπα εγώ σαρκαστικά. «Μα δεν είναι θέμα χρημάτων. Ή μάλλον, ακόμα και αν υπάρχει θέμα χρημάτων, αυτό δεν είναι το βασικό. Το βασικό στην Αγγλία είναι ο έλεγχος. Θα έρθει η επιθεώρηση εργασίας και μαύρο φίδι που σε έφαγε αν πιαστείς να κόβεις το ρεπό του αλλουνού ή να μην του πληρώνεις τις υπερωρίες. Μαύρο φίδι.»

«Να το λοιπόν. Αυτοί έχουν το μαύρο φίδι της επιθεώρησης εργασίας, κι εμείς έχουμε το λευκό βατράχι του ΣΔΟΕ με τους αδιάφθορους. Γι αυτό και έχουμε βρει προκοπή», είπε η κυρία Καίτη και βάλαμε όλοι τα γέλια.

«Εγώ δεν το πιστεύω ότι είμαστε τόσο χάλια», σχολίασε η εξαδέλφη. «Δεν ακούτε τις ειδήσεις; Κάθε μέρα όλο και κάποιον πιάνουνε, τον ένα για χρέη προς το δημόσιο, τον άλλο επειδή δεν έκοψε απόδειξη, επειδή το ένα, επειδή το άλλο».

«Πού να σας λέω», την έκοψε η κυρία Καίτη, «που είχα πάει τις προάλλες στο ΙΚΑ για κάποια βεβαίωση. Τέλη του μήνα και έληγε κάποια προθεσμία για όσους παίρνουν αναπηρική σύνταξη. Κολυμπήθρα του Σιλωάμ είχε γίνει. Στην ουρά για να περάσουν από γιατρό. Χαμός, πραγματικά χαμός...»

«Μα τέτοια σκληρότητα να ζητούν από άρρωστους άνθρωπους να στήνονται στην ουρά για μια βεβαίωση...» σχολίασε θλιμμένα η μητέρα της εξαδέλφης που είναι Ινδουίστρια.

«Άρρωστοι άνθρωποι δεν λες τίποτα. Αφού στην είσοδο του ΙΚΑ, είχε στηθεί πανηγύρι. Είχανε μαζευτεί πραγματευτάδες και ο ένας νοίκιαζε αναπηρικές πολυθρόνες, και πατερίτσες, ο άλλος πούλαγε ορθοπεδικά κολλάρα, ο άλλος μπαστούνια για τυφλούς. Λαϊκή αγορά.»

Μείναμε να την κοιτάζουμε.

«Όπως σας το λέω. Ερχότανε ο άλλος καμαρωτός με τη μοτοσυκλέτα του, την κλείδωνε, πήγαινε παραπέρα, μπροστά στα μάτια του σεκιούριτι και των υπαλλήλων, νοίκιαζε κάποια πατερίτσα που να του ταιριάζει, και μετά έμπαινε στο ΙΚΑ να τον κοιτάξει ο γιατρός να τον λυπηθεί. Έβγαινε, επέ-



στρεφε την πατερίτσα, καβαλούσε την μοτοσυκλέτα του, και μην τον είδατε. Παστρικές δουλειές. Και μετά μου λέτε για επιθεώρηση εργασίας. Οι μεν θα κάνουν ότι επιθεωρούν, και οι δε θα κάνουν ότι επιθεωρούνται. Τρεις θα λαλούν και δυο θα χορεύουν.»

Γελάσαμε όλοι λίγο και μετά μείναμε σιωπηλοί επιστρέφοντας ο καθένας στις ασχολίες του.

Λοιπόν, το σχόλιο που ήθελα να κάνω σχετικά με το editorial του κυρίου Αρχισυντάκτη, στο τελευταίο τεύχος του περιοδικού, είναι το εξής. Θα το διαβάσατε, υποθέτω. Αν όχι σας προτρέπω να το κάνετε.

Εγραφε, λοιπόν, ο κύριος Αρχισυντάκτης για τη βία και σημείωνε ότι η «λογική του 'καταδικάζουμε τη βία από όπου κι αν προέρχεται', (...) είναι εντελώς ανιστόρητη κι άκρως επικίνδυνη', επειδή 'εξισώνει τον εκμεταλλευτή με τον εκμεταλλευόμενο. (...) Το αν η χρήση της (βίας) είναι δίκαιη ή άδικη (...) συνδέεται με το αν αυτός που τη χρησιμοποιεί είναι εκμεταλλευτής ή εκμεταλλευόμενος. Εξίσου σημαντικό ρόλο παίζει το γιατί επιστρατεύεται η βία. Διότι αν ένας εκμεταλλευόμενος αξιοποιεί τη βία μόνο και μόνο για να πάρει τη θέση του εκμεταλλευτή, τότε η χρήση της είναι και πάλι άδικη.'

Αναρωτιόμουν. Πώς μπορούμε να γνωρίζουμε αν ένας εκμεταλλευόμενος αξιοποιεί την βία *μόνο και μόνο* για να πάρει την θέση του εκμεταλλευτή; Τι θα πει «μόνο και μόνο»; Πώς μπορούμε να είμαστε σίγουροι; Εχουμε πρόσβαση στις μύχιες σκέψεις του εκμεταλλευόμενου που εξεγείρεται; Γνωρίζουμε το μέλλον;

Όχι. Τίποτα από τα δύο. Ψάξτε το λίγο. Διαβάστε, ας πούμε, για τη σφαγή δεκάδων χιλιάδων αμάχων κατά την Άλωση της Τριπολιτσάς, ή για τις εκτελέσεις κατά την περίοδο της τρομοκρατίας τον Ιούλιο του 1793 στην επαναστατημένη Γαλλία. Προσπαθήστε να τα εξηγήσετε με όρους εκμεταλλευτή και εκμεταλλευόμενου. Θα δείτε ότι δεν γίνεται. Θα πρέπει να εισάγετε νέους όρους όπως εκδίκηση, εθνοκάθαρση, ισορροπίες εξουσίας κ.λπ. κ.λπ. κ.λπ.

Αγαπητοί μου,

Αν το σκεφτούμε «καθαρά και συστηματικά», θα δούμε ότι ποτέ δεν μπορούμε να ξέρουμε τι είχε πραγματικά στο νου του ο κάθε εκμεταλλευόμενος που εξεγείρεται. Ποτέ δεν μπορούμε να ξέρουμε την πορεία που θα έχουν οι εξελίξεις. Και ποτέ δεν είναι τόσο ξεκάθαρο και σαφές ποιος είναι ο εκμεταλλευτής και ποιος ο εκμεταλλευόμενος.

Όλα αυτά μας οδηγούν σε ένα μόνο συμπέρασμα: Αν θέλουμε να αποφύγουμε το fallacy της μη διάκρισης επιπέδων (για το οποίο έχω γράψει και ξαναγράψει) κι αν δεν θέλουμε να πέσουμε στη γνωστή λούμπα να νομίζουμε ότι κατανοούμε την εσωτερική λογική των γεγονότων στα οποία συμμετέχουμε, τότε *οφείλουμε* να καταδικάζουμε τη βία από όπου και αν προέρχεται.

Εστω και αν φαντάζουμε αφελείς, σαν τον δήμαρχο της Αθήνας.

Εστω και αν σε κάποιες περιπτώσεις χρειαστεί ωστόσο να πάρουμε τα όπλα.

Δεν χρειάζεται να είσαι υπέρ της βίας για να πάρεις το όπλο. Η αντίφαση είναι trivial. Μη αυστηρή.

Θα επιστρέψω σε αυτό την άλλη φορά. Τώρα θα μου επιτρέψετε να ετοιμαστώ για ούζο, πολίτικους μεζέδες, ηλιοβασίλεμα, και καζάν ντιμπί.

Σας ασπάζομαι,
Θείος Ακάκιος

Skill: Intermediate
Tags: VGA, scanlines, VSYNC, HSYNC, refresh rate, pixel clock, racing the beam

By Spir@lEvolut10n



Έξοδος στη μεγάλη οθόνη [μέρος 1]

Μη νομίζετε ότι αναφερόμαστε στον κινηματογράφο – δεν έχουμε ψωνιστεί τόσο πολύ ακόμα. Απλά, στο βαθμό που ασχολούμαστε με ταπεινά LED matrix και με μικροσκοπικές οθόνες LCD, η οθόνη της τηλεόρασης φαντάζει γιγαντιαία. Πώς θα σας φαινόταν αν το επόμενο project σας απεικόνιζε δεδομένα στην τηλεόραση; Κι όταν μιλάμε για τηλεοράσεις δεν εννοούμε εκείνες τις παλιές, που θυμίζουν μικρές ντουλάπες. Η κατασκευή που έχουμε κατά νου θα παράγει σήμα VGA και θα συνδέεται σε κάθε σύγχρονη οθόνη, όπως και σε πολλούς βιντεοπροβολείς (και ίσως έτσι να φτάσουμε και στην πραγματικά μεγάλη οθόνη ;))

Όσοι ασχολούνται αρκετό καιρό με το Arduino έχουν ακούσει και για τη βιβλιοθήκη tv-out. Εξάλλου, ο συνάδελφος και φίλος GiannouG φρόντισε να παρουσιάσει ένα μέρος των δυνατοτήτων της από πολύ νωρίς — μιλάμε για τα τεύχη 002 και 003 του περιοδικού! Η εν λόγω βιβλιοθήκη προσφέρει πολλές ευκολίες: Διαθέτει συναρτήσεις για τη σχεδίαση βασικών σχημάτων, την εμφάνιση αποθηκευμένων bitmaps και την προβολή κειμένου. Εν ολίγοις, αν σκοπεύουμε να συνδέσουμε το Arduino σε κάποια παλιά τηλεόραση, η βιβλιοθήκη tv-out αποτελεί την ιδανική λύση. Μήπως αυτό σημαίνει ότι ο στόχος μας έχει επιτευχθεί ήδη και το υπόλοιπο άρθρο είναι περιττό; (Σ.τ.Α. Λέτε να είναι πράγματι περιττό; :P)

Αν και η βιβλιοθήκη tv-out είναι πραγματικά αξιόλογη, θέτει μερικούς ενοχλητικούς περιορισμούς. Καταρχάς, το σήμα που συνθέτει είναι τύπου Composite (PAL ή NTSC). Αντίστοιχη είσοδο δεν θα βρούμε σε κανένα monitor υπολογιστή, ενώ στις σύγχρονες τηλεοράσεις παρουσιάζονται διάφορα προβλήματα με το σήμα που παράγει η βιβλιοθήκη[1]. Έτσι, το σήμα της βιβλιοθήκης tv-out προορίζεται σχεδόν αποκλειστικά για τις παλιές, χοντροκομμένες τηλεοράσεις. Επιπρόσθετα, η εικόνα που παράγεται είναι μονόχρωμη (λευκό και μαύρο, χωρίς τις ενδιάμεσες διαβαθμίσεις) κι έχει μέγιστη ανάλυση που φτάνει μόλις τα 128x96 pixels.

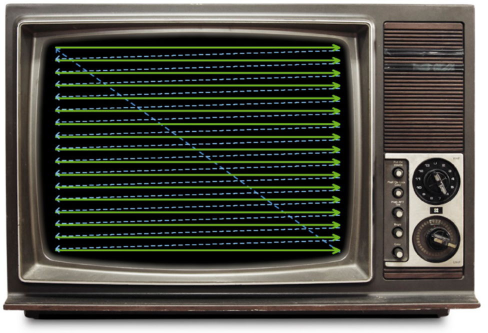
Ε, λοιπόν, αν είναι να βγούμε στο γυαλί, θα το κάνουμε πολύ πιο εντυπωσιακά[2]! Για το σκοπό αυτό θα ασχοληθούμε με το σήμα VGA, θα εξετάσουμε τη δομή του και θα δούμε πώς μπορούμε να το συνθέσουμε με τη βοήθεια ενός μικροελεγκτή. Το εν λόγω σήμα θα είναι κι αυτό αναλογικό, αλλά υποστηρίζεται από όλες σχεδόν τις οθόνες υπολογιστή και τις σύγχρονες τηλεοράσεις. Η παρουσίασή μας θα χωριστεί σε δύο μέρη: Στο πρώτο θα αποκτήσουμε την απαραίτητη εξοικείωση με το σήμα VGA και η κατασκευή που θα παρουσιάσουμε θα προβάλλει μόνο κείμενο. Συγκεκριμένα, θα έχει τη δυνατότητα να εμφανίζει 24 γραμμές μονόχρωμου κειμένου, με 32 χαρακτήρες ανά γραμμή. Στο δεύτερο μέρος θα παρουσιάσουμε μια πιο εξελιγμένη εκδοχή της κατασκευής, η οποία θα μπορεί να προβάλλει εικόνα και κείμενο με 256 χρώματα. Σημειώστε ότι και στις δύο κατασκευές η ανάλυση της εικόνας θα φτάνει στα 256x240 pixels! Για να πετύχουμε μεγαλύτερη ανάλυση θα χρειαστούμε λίγη περισσότερη ισχύ από εκείνη που παρέχει το Arduino. Μην τρέχει το μυαλό σας σε κανένα εξωτικό τσιπάκι. Θα χρησιμοποιήσουμε τον (φτηνό) μικροελεγκτή AVR, στον οποίο στηρίζεται και το Arduino. Μόνο που εμείς θα τον τοποθετήσουμε σ’ ένα απλούστερο κύκλωμα και θα τον λειτουργούμε στα 20MHz. Ευτυχώς δηλαδή που στο προηγούμενο τεύχος, το 025, μάθαμε τα πάντα για τον προγραμματισμό και το χειρισμό των AVR, έξω από το περιβάλλον του Arduino.

Αναλογικά σήματα εικόνας

Γνωρίζετε πώς λειτουργούσαν οι παλιές τηλεοράσεις με την καθοδική λυχνία; Το εν λόγω εξάρτημα, που οι παλαίμαχοι ηλεκτρονικοί ονόμαζαν και "πυροβόλο", εξέπεμπε μια δέσμη ηλεκτρονίων. Αυτή η δέσμη λειτουργούσε σαν ένα λεπτεπίλεπτο πινέλο, που σάρωνε την επιφάνεια της οθόνης και σχημάτιζε την εικόνα. Η σάρωση ξεκινούσε από το πάνω αριστερό άκρο της οθόνης (όπως την κοιτάμε), ακολουθούσε μια πορεία ζιγκ-ζαγκ κι έφτανε μέχρι το κάτω δεξιό άκρο. Με αυτόν τον τρόπο η εικόνα σχηματιζόταν γραμμή προς γραμμή. Φυσικά, η σάρωση γινόταν με μεγάλη ταχύτητα και δεν αντιλαμβανόμασταν τη σχεδίαση των ξεχωριστών γραμμών, αλλά

1. Οι σύγχρονες τηλεοράσεις ψηφιοποιούν οποιοδήποτε αναλογικό σήμα δέχονται, πριν το επεξεργαστούν και το προβάλλουν. Ως εκ τούτου, το σήμα που παράγει η βιβλιοθήκη tv-out δεν είναι ιδιαίτερα ευπρόσδεκτο στις σύγχρονες τηλεοράσεις! Το σήμα της βιβλιοθήκης παράγεται με PWM (δεν είναι πραγματικά αναλογικό) και προκαλεί «σύγχυση» στους ADC των τηλεοράσεων.

2. Στ’ αλήθεια δεν έχουμε ψωνιστεί ;)



Εικόνα 1
Στις τηλεοράσεις με την καθοδική λυχνία (το πυροβόλο!) η εικόνα σχηματιζόταν με τη βοήθεια μιας δέσμης ηλεκτρονίων, η οποία σάρωνε την επιφάνεια της οθόνης. Η σάρωση ξεκινούσε από πάνω αριστερά και ακολουθώντας μια πορεία ζιγκ-ζαγκ έφτανε κάτω δεξιά. Έτσι, η εικόνα σχηματιζόταν γραμμή προς γραμμή! Ανάμεσα στη σχεδίαση κάθε γραμμής μεσολαβούσε ένα «νεκρό» διάστημα, κατά το οποίο η δέσμη επέστρεφε στην αριστερή πλευρά της οθόνης και ετοιμαζόταν για τη νέα γραμμή. Ένα αντίστοιχο «νεκρό» διάστημα μεσολαβούσε και στις διαδοχικές σαρώσεις ολόκληρης της εικόνας, οπότε η δέσμη επέστρεφε από την κάτω δεξιά γωνία στην πάνω αριστερή.

ολόκληρη την εικόνα. Ο τρόπος λειτουργίας των παλιών τηλεοράσεων καθόρισε σε τεράστιο βαθμό την "εσωτερική δομή" των αναλογικών σημάτων εικόνας. Έτσι, σχεδόν όλα τα αναλογικά σήματα περιλαμβάνουν έναν παλμό που σηματοδοτεί την εκκίνηση της σάρωσης από το πάνω αριστερό άκρο (την έναρξη σχεδίασης ολόκληρου του κάδρου), έναν παλμό που σηματοδοτεί την εκκίνηση της σάρωσης για κάθε γραμμή και φυσικά την οπτική πληροφορία για κάθε γραμμή. Παρεμπιπτόντως, να πούμε ότι αυτές οι γραμμές ονομάζονται και scanlines.

Το σήμα VGA, παρόλο που χρησιμοποιείται ευρύτατα μέχρι σήμερα και υποστηρίζεται από οθόνες με εντελώς διαφορετική τεχνολογία, περιλαμβάνει τα ίδια ακριβώς "εξαρτήματα": Έναν παλμό κατακόρυφου συγχρονισμού, έναν παλμό οριζόντιου συγχρονισμού[3] και τα δεδομένα για τα διαδοχικά pixels που σχηματίζουν κάθε γραμμή. Όταν η οθόνη λαμβάνει τον παλμό κατακόρυφου συγχρονισμού (VSYNC), γνωρίζει ότι όλα τα δεδομένα που ακολουθούν μέχρι τον επόμενο παλμό VSYNC αφορούν στη σχεδίαση ενός ολόκληρου κάδρου. Έτσι, περιμένει πλέον τους παλμούς οριζόντιου συγχρονισμού (HSYNC) για να ξεκινήσει τη σχεδίαση κάθε γραμμής που απαρτίζει την εικόνα. Ανάμεσα στους παλμούς HSYNC μεταδίδονται οι πληροφορίες για την ένταση και το χρώμα των pixels, που περιλαμβάνει η εκάστοτε γραμμή.

Λεπτομέρειες και παραξενιές

Το πρότυπο VGA περιλαμβάνει πολλές παραλλαγές, καθεμία από τις οποίες προβλέπει διαφορετική ανάλυση για την εικόνα και διαφορετικό ρυθμό ανανέωσης (refresh rate) της εικόνας. Οι δικές μας κατασκευές θα στηριχτούν σε μία από τις παλιότερες παραλλαγές, που προβλέπει την ανάλυση 640x480 με ρυθμό ανανέωσης τα 60Hz. Αυτή η παραλλαγή είναι γνωστή και σαν "VGA industry standard". Θέλουμε να πιστεύουμε ότι μέχρι εδώ είναι τα πάντα κατανοητά. Και το θέλουμε πολύ αυτό, διότι στη συνέχεια θα δούμε ορισμένες παραξενιές που κρύβει το σήμα VGA.

Αν και η εικόνα (στο πρότυπο που εξετάζουμε) έχει ανάλυση 640x480, ανάμεσα σε δύο διαδοχικούς παλμούς VSYNC μεσολαβούν 525 παλμοί HSYNC. Με απλά λόγια, κάθε κάδρο περιλαμβάνει 525 γραμμές και όχι 480! Οι πρόσθετες γραμμές δεν προβάλλονται ποτέ στην οθόνη και έχουν προστεθεί για να καθυστερεί η σχεδίαση του επόμενου κάδρου. Αναρωτιέστε σε τι ωφελεί αυτή η καθυστέρηση; Η απάντηση βρίσκεται στον τρόπο λειτουργίας των παλιών τηλεοράσεων. Σε αυτές, η εκτροπή της δέσμης των ηλεκτρονίων γινόταν με τη βοήθεια ειδικών πηνίων, που λειτουργούσαν

3. Ονομάζονται VSYNC (Vertical Synchronization) και HSYNC (Horizontal Synchronization).

σαν ηλεκτρομαγνήτες[4]. Η συμπεριφορά των πηνίων, όμως, παρουσιάζει αδράνεια. Αυτό σημαίνει ότι κάθε φορά που η δέσμη ηλεκτρονίων φτάνει στο κάτω δεξιό μέρος της οθόνης, δεν φρενάρει ακαριαία. Επιπρόσθετα, όταν επιστρέφει στο πάνω αριστερό άκρο της οθόνης (κάτι που επίσης απαιτεί χρόνο) ούτε και τότε φρενάρει μονομιάς. Οι πρόσθετες γραμμές, λοιπόν, που περιλαμβάνονται στο κάδρο αλλά δεν προβάλλονται ποτέ, έχουν τοποθετηθεί για να προλαβαίνει η δέσμη να επιστρέψει στην αρχική της θέση. Ουσιαστικά, αυτές οι πρόσθετες γραμμές αποτελούν απομεινάρι του παρελθόντος, που διατηρήθηκε στην τυποποίηση του σήματος για λόγους συμβατότητας. Κάπου εδώ, θα σκεφτεί κανείς ότι η αδράνεια των πηνίων θα έπρεπε να επηρεάζει κάθε απότομη μεταβολή στην κίνηση της δέσμης. Επομένως, θα έπρεπε να υπήρχε και μια καθυστέρηση κατά την επιστροφή της δέσμης από τη δεξιά άκρη της οθόνης στην αριστερή (για τη σχεδίαση της επόμενης γραμμής). Έτσι είναι! Γι' αυτό το λόγο, αν και η ανάλυση της εικόνας στο VGA industry standard είναι 640x480, κάθε γραμμή περιλαμβάνει 800 pixels και όχι 640. Τα πρόσθετα pixels δεν εμφανίζονται ποτέ και υπάρχουν μόνο για να προλάβει η δέσμη να φτάσει στη θέση εκκίνησης της επόμενης γραμμής. Τελικά, ένα κάδρο στο VGA industry standard περιλαμβάνει 525 γραμμές και καθεμία από αυτές περιλαμβάνει 800 pixels. Στους ακόλουθους πίνακες φαίνονται όλα τα επιμέρους χρονικά διαστήματα που απαρτίζουν κάθε γραμμή και κάθε κάδρο.

Σε κάθε γραμμή:		
front porch	16 pixels	Ο απαραίτητος χρόνος για την επιστροφή της δέσμης στην αρχή της γραμμής. Ονομάζεται και **horizontal blanking period.**
horizontal sync	96 pixels	
back porch	48 pixels	
active video	640 pixels	Ορατά pixels
Σύνολο 800 pixels ανά γραμμή		

Πίνακας 1

Σε κάθε κάδρο:		
front porch	10 γραμμές	Ο απαραίτητος χρόνος για την επιστροφή της δέσμης στην αρχή τους κάδρου. Ονομάζεται και **vertical blanking period.**
vertical sync	2 γραμμές	
back porch	33 γραμμές	
active video	480 γραμμές	Ορατές γραμμές
Σύνολο 525 γραμμές ανά κάδρο		

Πίνακας 2

4. Γι' αυτό και πολλές φορές, όταν η εικόνα έμοιαζε ελαφρώς παραμορφωμένη, κάναμε μια απομαγνήτιση των πηνίων (degauss). Με αυτόν τον τρόπο, ήταν σαν να κάνουμε μια «επανεκκίνηση» στη μαγνητική συμπεριφορά των πηνίων. Ωραίες εποχές!

Περισσότερες λεπτομέρειες

Τα καλώδια VGA περιλαμβάνουν τουλάχιστον έξι διαφορετικούς αγωγούς. Σε έναν από αυτούς ταξιδεύουν οι παλμοί οριζόντιου συγχρονισμού και σε έναν άλλο οι παλμοί κατακόρυφου συγχρονισμού. Για το χρώμα κάθε pixel χρησιμοποιούνται τρεις διαφορετικοί αγωγοί. Ο καθένας από αυτούς μεταφέρει την τιμή για ένα από τα βασικά χρώματα (κόκκινο, πράσινο και μπλε). Με λίγα λόγια, οι δύο αγωγοί μεταφέρουν τα βοηθητικά σήματα (παλμούς συγχρονισμού) ενώ οι άλλοι τρεις μεταφέρουν την πληροφορία για τη σύνθεση της εικόνας (το χρώμα κάθε διαδοχικού pixel). Τέλος, ένας έκτος αγωγός συνδέεται στο GND της οθόνης και της εκάστοτε πηγής σήματος, ώστε τα δύο κυκλώματα να έχουν την ίδια τάση αναφοράς. Για να κατανοήσετε καλύτερα πώς μεταφέρεται η εικόνα, ρίξτε μια ματιά στην εικόνα 2.

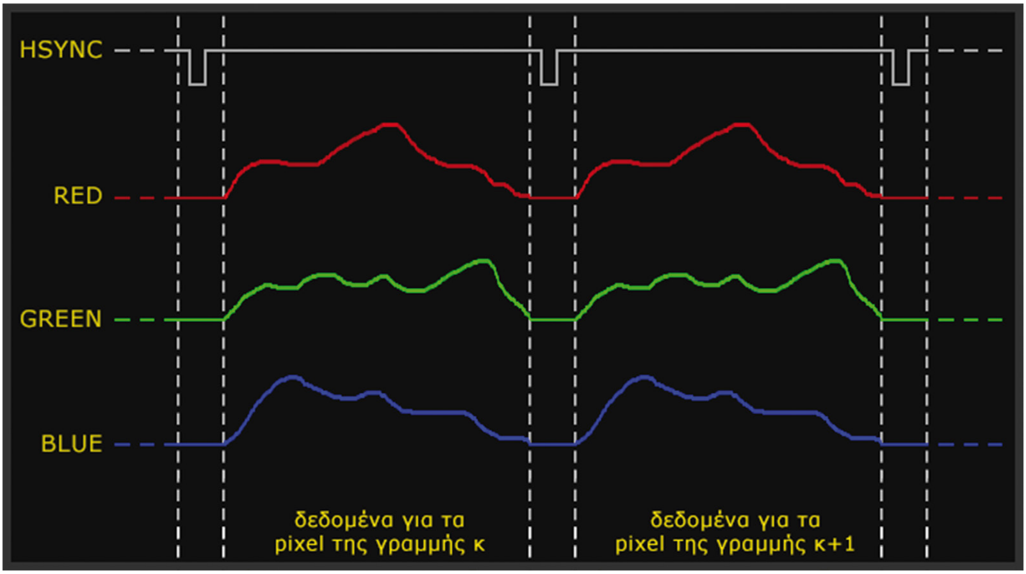
Πριν προχωρήσουμε, αξίζει να πούμε δυο λόγια για τη φύση των επιμέρους σημάτων που απαρτίζουν το σήμα VGA. Οι παλμοί οριζόντιου και κατακόρυφου συγχρονισμού είναι ψηφιακοί και ειδικά στην περίπτωση του VGA industry standard είναι και αντεστραμμένοι. Αυτό σημαίνει ότι οι αντίστοιχοι αγωγοί βρίσκονται συνεχώς στην κατάσταση HIGH (5V) κι όταν ενεργοποιείται ο παλμός μεταβαίνουν στην κατάσταση LOW (0V). Η τιμή κάθε χρώματος μεταδίδεται σαν μια τάση στην περιοχή από 0 έως 0,7V. Τα 0V αντιστοιχούν σε μηδενική φωτεινότητα για το αντίστοιχο χρώμα και τα 0,7V αντιστοιχούν σε πλήρη ένταση.

Εξετάζοντας τους πίνακες που συναντήσαμε νωρίτερα, όπως και την εικόνα 2, είναι πιθανό να αναρωτηθείτε το εξής: Πόσο διαρκεί καθένα από τα τμήματα που αναφέρονται; Πόσος χρόνος χρειάζεται για τη μετάδοση των δεδομένων ενός pixel και κατ' επέκταση μιας ολόκληρης γραμμής; Αυτό το ερώτημα μπορεί να μοιάζει αδιάφορο, αλλά δεν είναι. Για να μπορεί μια οθόνη να προβάλλει σωστά το σήμα που δέχεται, πρέπει να ξέρει πότε και τι να περιμένει. Με άλλα λόγια, οι παλμοί συγχρονισμού και τα δεδομένα για κάθε γραμμή πρέπει να φτάνουν σε συγκεκριμένες χρονικές αποστάσεις αναμεταξύ τους και, φυσικά, όλα τα δεδομένα πρέπει να μεταδίδονται με σταθερό ρυθμό. Γι' αυτό το λόγο, κάθε παραλλαγή του προτύπου VGA περιλαμβάνει στις προδιαγραφές της μια κατάλληλη συχνότητα για το λεγόμενο pixel clock. Σε κάθε παλμό αυτού του "ρολογιού" μεταδίδεται η πληροφορία για ένα pixel.

Στην περίπτωση που εξετάζουμε (VGA industry standard) το pixel clock έχει οριστεί στα 25,175MHz. Αυτή η συχνότητα δεν επιλέχθηκε τυχαία. Τα 800 pixels που απαρτίζουν μια ολόκληρη γραμμή μεταδίδονται σε 800 παλμούς του "ρολογιού". Με συχνότητα 25,175MHz, ο κάθε παλμός διαρκεί (1/25,175)μSec κι άρα τα 800pixel μεταδίδονται σε 31,775μSec. Αυτό σημαίνει ότι οι πληροφορίες για 525 γραμμές (ένα πλήρες κάδρο) μεταδίδονται σε 525 * 31,775μSec = 16,683mSec. Επομένως, το refresh rate (ο ρυθμός ανανέωσης της εικόνας) ανέρχεται στα (1 / 16,683)KHz = 59,84Hz. Τελικά, διαπιστώνουμε ότι το pixel clock είναι ορισμένο με τέτοιον τρόπο, ώστε για μια δοσμένη ανάλυση να επιτυγχάνεται ένας συγκεκριμένος ρυθμός ανανέωσης.

Ζήτημα χρόνου

Σε αυτό τα στάδιο γνωρίζουμε αρκετά καλά τη μορφή των επιμέρους σημάτων που πρέπει να δημιουργήσουμε, όπως και τον μεταξύ τους χρονικό συσχετισμό. Μήπως αυτό σημαίνει ότι μπορούμε ν' αρχίσουμε να γράφουμε κώδικα; Μάλλον όχι! Βλέπετε, το pixel clock για το πρότυπο που θέλουμε να υλοποιήσουμε είναι στα 25,175MHz, ενώ ο μικροελεγκτής μας μπορεί να λειτουργήσει το πολύ στα 20MHz.



Εικόνα 2

Τα χρωματικά δεδομένα για τα pixels μιας γραμμής (scanline) μεταδίδονται μετά από τον παλμό HSYNC, που σηματοδοτεί την «έναρξη» της γραμμής. Με αυτό τον τρόπο (γραμμή προς γραμμή) μεταδίδονται τα δεδομένα για τα pixels ολόκληρου του κάδρου. Παρατηρείστε ότι αμέσως πριν και αμέσως μετά από κάθε παλμό HSYNC εμφανίζεται ένα χρονικό διάστημα κατά το οποίο δεν μεταδίδεται καμία πληροφορία. Τα δύο αυτά διαστήματα (front porch και back porch) απαρτίζουν το λεγόμενο horizontal blanking time.

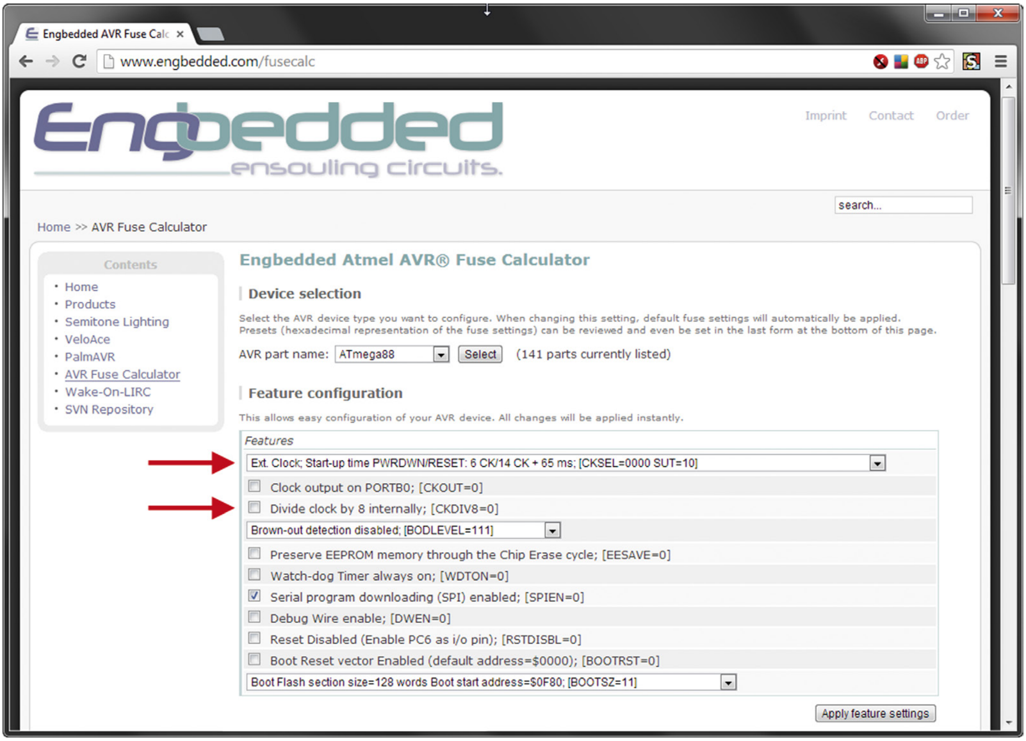
Και πριν σκεφτείτε ότι το πρόβλημα λύνεται με λίγο overclocking, επιτρέψτε μας να σας επισημάνουμε μια λεπτομέρεια: Οι μικροελεγκτές της οικογένειας AVR μπορούν να ανανεώνουν την κατάσταση των ακροδεκτών τους στη μισή συχνότητα από εκείνη του πυρήνα τους. Με άλλα λόγια, αν χρονίσουμε τον μικροελεγκτή στα 20MHz, οι ακροδέκτες θα μπορούν να αλλάζουν κατάσταση με μέγιστη συχνότητα τα 10MHz. Τελικά, αν μετατρέψουμε έναν AVR σε πηγή σήματος VGA, το μέγιστο pixel clock που μπορούμε να πετύχουμε αγγίζει μόλις τα 10MHz. Με τόσο χαμηλό ρυθμό μετάδοσης, ο μικροελεγκτής μας δεν προλαβαίνει να μεταδώσει δεδομένα για 800 pixels, μέσα στο χρονικό διάστημα που προβλέπεται για τη σάρωση μιας γραμμής. Το ίδιο ισχύει και για τα 640 pixels που απαρτίζουν το ορατό τμήμα κάθε γραμμής.

Αριθμητικό μαγείρεμα

Κάπου εδώ αρχίζουν οι πατέντες! Για να "κλειδώσει" η εκάστοτε οθόνη στο σήμα μας, πρέπει να ακολουθήσουμε τις προδιαγραφές κάποιου προτύπου. Όπως έχουμε αναφέρει, επιλέξαμε το VGA industry standard γιατί έχει τη χαμηλότερη ανάλυση (640x480) και το μικρότερο refresh rate (60Hz)[5]. Έτσι, ερχόμαστε αντιμέτωποι με το εξής ερώτημα: Πώς θα μπορέσουμε να συμμορφωθούμε στις προδιαγραφές του συγκεκριμένου προτύπου;

Οι οθόνες αδιαφορούν παγερά για το περιεχόμενο των pixels. Είτε έχουν όλα το ίδιο χρώμα, είτε εναλλάσσονται τυχαία, είτε ακολουθούν κάποιο μοτίβο, για την οθόνη δεν αλλάζει τίποτα. Σκεφτείτε το λίγο αυτό. Στο χρόνο που ο μικροελεγκτής μπορεί να στείλει δεδομένα για ένα μόνο pixel, η οθόνη θα περιμένει δεδομένα για περισ-

5. Επομένως είναι και το ευκολότερο στην υλοποίηση.



Εικόνα 3

Για την κατασκευή μας απαιτούνται δύο κρίσιμες ρυθμίσεις στα fuse bytes του μικροελεγκτή. Αφενός πρέπει να επιλέξουμε τη χρήση εξωτερικού ρολογιού (external clock) και αφετέρου πρέπει να απενεργοποιήσουμε την εσωτερική διαίρεση του ρολογιού (Divide clock by 8 internally). Για την πραγματοποίηση των ρυθμίσεων μπορούμε να χρησιμοποιήσουμε το εργαλείο avrdude με τις κατάλληλες παραμέτρους, που θα πάρουμε από τη δικτυακή εφαρμογή στο `www.engbedded.com/fusecalc`.

σότερα pixels και πολύ απλά, θα νομίζει ότι έχουν όλα την ίδια τιμή. Με άλλα λόγια, το ένα pixel που στέλνει ο μικροελεγκτής θα αντιστοιχεί σε περισσότερα pixels της οθόνης. Κατά κάποιον τρόπο, λοιπόν, σε άλλη ανάλυση θα εργάζεται ο μικροελεγκτής και σε μια διαφορετική θα λειτουργεί η οθόνη! Τελικά, ποιο είναι το μέγιστο πλήθος pixels που προλαβαίνει να στείλει ο μικροελεγκτής μας; Την απάντηση θα μας δώσει η απλή μέθοδος των τριών:

```
Σε μία γραμμή (31,775μSec) και στα 25,175MHz εκπέμπονται 800 pixels
Σε μία γραμμή (31,775μSec) και στα 10,000MHz εκπέμπονται x pixels
-------------------------------------------------------------------
x = 800 * 10 / 25,175 = 317,77 pixels
```

Τελικά, ένας πρώτος περιορισμός για τη συμμόρφωση στο VGA industry standard είναι να σχηματίζουμε scanlines των 317,77 pixels και μάλιστα με το μέγιστο ρυθμό που επιτρέπουν οι ακροδέκτες του μικροελεγκτή. Βέβαια, τα pixels δεν μπορούν παρά να έχουν ακέραιο πλήθος. Οπότε το 317,77 θα γίνει κατά προσέγγιση 318. Τι

άλλο χρειάζεται για να κλειδώσει η οθόνη σε ένα σήμα; Μα, φυσικά, οι παλμοί οριζόντιου και κατακόρυφου συγχρονισμού. Ας εξετάσουμε τον παλμό HSYNC. Πόσο θα διαρκεί αυτός ο παλμός για τα δεδομένα του μικροελεγκτή;

```
Στα 25,175MHz ο παλμός HSYNC διαρκεί 96 pixels
Στα 10,000MHz ο παλμός HSYNC διαρκεί x pixels
----------------------------------------------
x = 96 * 10 / 25,175 = 38,13 pixels
```

Επομένως, για να ακολουθούμε το VGA industry standard πρέπει να σχηματίζουμε παλμούς HSYNC με διάρκεια ισοδύναμη των 38 pixels. Οι παλμοί VSYNC διαρκούν για ολόκληρες γραμμές και ο σχηματισμός τους είναι ευκολότερος. Θα αναφερθούμε αργότερα σε αυτούς.

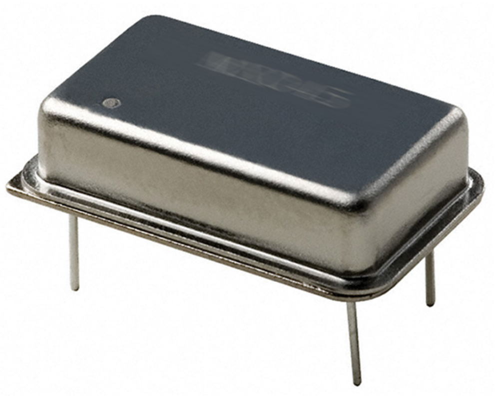
Με τον ίδιο τρόπο σκέψης μπορούμε να υπολογίσουμε τη διάρκεια για όλα τα επιμέρους τμήματα μιας γραμμής (front porch, back porch και active video). Αν κάνουμε τους υπολογισμούς, θα δούμε ότι το ορατό τμήμα κάθε γραμμής θα έχει μήκος 254,22 pixels. Εδώ θα κάνουμε πάλι μια πονηριά! Το 254 είναι κοντά στο (πολύ βολικό) 256. Μήπως θα μπορούσαμε να θεωρήσουμε ότι το ορατό τμήμα των γραμμών έχει μήκος 256 pixels; Κανείς δεν μας εμποδίζει! Τα σήματα που περιγράφουν το χρώμα κάθε pixel είναι αναλογικά κι αν τα "ξεχειλώσουμε" ελαφρώς, η οθόνη δεν θα ενοχληθεί ιδιαίτερα. Σε αυτή την περίπτωση, όμως, θα πρέπει να ξεκλέψουμε ισάριθμο πλήθος pixels από το back porch ή το front porch, ώστε η συνολική διάρκεια της γραμμής να μην ξεφύγει από το όριο που θέτει το πρότυπο. Κάπως έτσι, καταλήξαμε στον ακόλουθο πίνακα:

Σε κάθε γραμμή (για τον μικροελεγκτή):		
front porch	6 pixels	Ο απαραίτητος χρόνος για την επιστροφή της δέσμης στην αρχή της γραμμής. Ονομάζεται και "horizontal blanking period".
horizontal sync	38 pixels	
back porch	18 pixels	
active video	256 pixels	Ορατά pixels
Σύνολο 318 pixels ανά γραμμή		

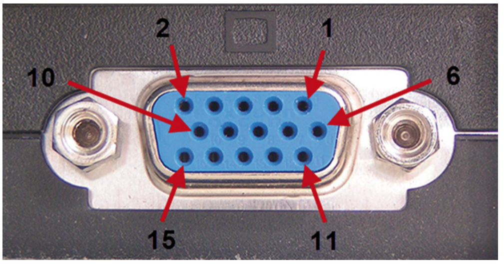
Πίνακας 3

Κι άλλη έκπτωση

Ο μικροελεγκτής προλαβαίνει (έστω κι οριακά) να σχηματίσει όλα τα επιμέρους τμήματα που απαρτίζουν μια γραμμή, μέσα στο χρονικό περιθώριο που προβλέπει το VGA industry standard. Επομένως, τίποτα δεν τον εμποδίζει να σχηματίσει όλες τις γραμμές (480). Αυτό σημαίνει ότι η σύνθεση του παλμού VSYNC, που διαρκεί για δύο ολόκληρες γραμμές (scanlines), δεν πρόκειται να μας δυσκολέψει. Πριν προχωρήσουμε, όμως, πρέπει να κάνουμε μια μικρή επισήμανση σε σχέση με την ανάλυση. Όπως είδαμε παραπάνω, το ορατό τμήμα κάθε γραμμής θα διαθέτει 256 pixels. Αν



Εικόνα 4
Ο ταλαντωτής που θα χρησιμοποιήσουμε σαν system clock για την κατασκευή μας. Τα συγκεκριμένα εξαρτήματα διαθέτουν τέσσερις ακροδέκτες, οι δύο από τους οποίους προορίζονται για την τροφοδοσία και ο τρίτος για την έξοδο του παλμού του ρολογιού. Ο τέταρτος ακροδέκτης δεν χρησιμοποιείται και είναι απλά… διακοσμητικός.



Εικόνα 5
Η υποδοχή VGA είναι γνωστή και σαν D-SUB 15. Οι επαφές 1, 2 και 3 μεταφέρουν τα χρωματικά δεδομένα για το κόκκινο, το πράσινο και το μπλε. Οι επαφές 13 και 14 μεταφέρουν τους παλμούς HSYNC και VSYNC αντίστοιχα. Χρήσιμες για το κύκλωμά μας είναι και οι επαφές 5, 6, 7, 8 και 10, που συνδέονται στο GND του κυκλώματος της οθόνης.

ο μικροελεγκτής σχηματίζει εικόνες με 480 ξεχωριστές γραμμές, η ανάλυση των εικόνων που θα προβάλλονται στην οθόνη θα είναι 256x480. Δεν ξέρουμε αν το αντιλαμβάνεστε, αλλά αυτή η ανάλυση οδηγεί σε αλλοπρόσαλλο PAR[6]: Τα pixels που θα βλέπουμε στην οθόνη θα έχουν παράξενο σχήμα και συγκεκριμένα θα είναι μακρόστενα — πολύ μακρόστενα! Αυτό το πρόβλημα ξεπερνιέται εύκολα, αν αποφασίσουμε ότι ο μικροελεγκτής θα προβάλλει εικόνες με ανάλυση 256x240. Με αυτόν τον τρόπο, τα pixels που θα σχηματίζονται στην οθόνη θα είναι περισσότερο τετραγωνισμένα. Για να πετύχουμε αυτήν την κατακόρυφη ανάλυση (240 γραμμές) χωρίς να ξεφύγουμε από τις απαιτήσεις του προτύπου (480 scanlines), ο μικροελεγκτής θα στέλνει κάθε γραμμή της εικόνας εις διπλούν. Με άλλα λόγια, ανά δύο scanlines θα εκπέμπονται τα ίδια ακριβώς pixels.

Προγραμματιστικές προδιαγραφές

Στις τελευταίες δύο ενότητες καταλήξαμε σε έναν απαραίτητο συμβιβασμό. Δεχτήκαμε ότι ο μικροελεγκτής θα εκπέμπει εικόνες με ανάλυση 256x240 pixels, για να εξασφαλίσουμε ότι το σήμα θα συμμορφώνεται στις προδιαγραφές του VGA industry standard. Όμως, ούτε και τώρα μπορούμε να ξεκινήσουμε τη συγγραφή κώδικα. Πρέπει πρώτα να ξεκαθαρίσουμε μερικές λεπτομέρειες.

Προηγουμένως, υπολογίσαμε τη χρονική διάρκεια των γραμμών (και των επιμέρους τμημάτων) σε pixels. Για να γράψουμε κώδικα, πρέπει να υπολογίσουμε το χρόνο που αντιστοιχεί σε κάθε pixel. Με άλλα λόγια, πρέπει να βρούμε το διαθέσιμο χρόνο για τον υπολογισμό και την εκπομπή των δεδομένων ενός pixel. Στους υπολογισμούς που προηγήθηκαν, δεχτήκαμε ότι ο μικροελεγκτής λειτουργεί στη μέγιστη συχνότητα (20MHz) και

6. Τα αρχικά προκύπτουν από τις λέξεις Pixel Aspect Ratio. Το εν λόγω μέγεθος εκφράζει την αναλογία μεταξύ των φυσικών διαστάσεων ενός pixel.

άρα οι ακροδέκτες του μπορούν να μεταβάλλονται με το μέγιστο ρυθμό (10MHz). Εφόσον κάθε pixel πρέπει να μεταδίδεται σε έναν παλμό των 10MHz, ο επεξεργαστής πρέπει να υπολογίζει τα δεδομένα για κάθε pixel και να τα εμφανίζει σε κάποιον ακροδέκτη, μέσα σε αυτό ακριβώς το χρονικό διάστημα. Όμως, ένας παλμός των 10MHz διαρκεί όσο δύο παλμοί των 20MHz. Κατά συνέπεια, ο επεξεργαστής μπορεί να διαθέσει το πολύ δύο κύκλους του ρολογιού (CPU clock cycles) για κάθε pixel :/ Ζόρικα τα πράγματα!

Έχοντας κατά νου την παραπάνω αντιστοιχία, υπολογίσαμε τα CPU clock cycles που θα έχει στη διάθεσή του το πρόγραμμά μας, για κάθε τμήμα που απαρτίζει τις γραμμές (scanlines) του σήματος. Αργότερα, όταν υλοποιήσαμε το σχετικό πρόγραμμα, διαπιστώσαμε ότι η οθόνη "κλείδωνε" στο σήμα μας αλλά με μια μικρή παραμόρφωση. Για να εξαλειφθεί το πρόβλημα και να πάρουμε καθαρή και σταθερή εικόνα, τροποποιήσαμε ελαφρώς τους χρόνους και καταλήξαμε στις τιμές που φαίνονται παρακάτω:

Τμήμα γραμμής	Διάρκεια
front porch	11 CPU clock cycles
horizontal sync	76 CPU clock cycles
back porch	37 CPU clock cycles
active video	512 CPU clock cycles

Πίνακας 4

Προγραμματισμός και ρύθμιση

Τώρα έχουμε όλα τα απαραίτητα δεδομένα για να γράψουμε ένα πρόγραμμα που θα παράγει σήμα VGA. Θα χρειαστούμε όμως όρεξη — μαζί με άφθονη υπομονή. Βλέπετε, το πρόγραμμα θα πρέπει να λειτουργεί με τεράστια ακρίβεια. Οι παλμοί οριζόντιου και κατακόρυφου συγχρονισμού, όπως επίσης και οι περίοδοι εκπομπής δεδομένων για τα ορατά pixels κάθε γραμμής, πρέπει να ξεκινούν και να σταματούν με ακρίβεια σε επίπεδο CPU clock cycle. Ένα τέτοιο πρόγραμμα μπορεί να γραφτεί μόνο σε Assembly. Ο δικός μας κώδικας είναι γραμμένος για τον assembler της Atmel (avrasm2.exe), ενώ χωρίς κανένα παράπονο θα τον δεχτεί και ο assembler ονόματι avra, που τρέχει στο Linux.

Ο μικροελεγκτής που χρησιμοποιήσαμε είναι ένας ATmega88. Η μόνη διαφορά του συγκεκριμένου τσιπ σε σχέση με τον ATmega328 (τον επεξεργαστή του Arduino) είναι ότι διαθέτει 8KB μνήμης flash και όχι 32KB. Κατά τα άλλα πρόκειται για πανομοιότυπα τσιπ και μπορείτε να χρησιμοποιήσετε οποιοδήποτε από τα δύο. Ο κώδικας βρίσκεται στο `http://bit.ly/dh026vgatxtcon`. Στο πακέτο ZIP θα βρείτε το αρχείο HEX που πρέπει να γράψετε στο τσιπάκι, τα αρχεία με τον κώδικα σε Assembly και δύο αρχεία συμπερίληψης (include files). Το ένα από αυτά αφορά στον ATmega88 και το άλλο στον ATmega328. Αν θελήσετε να παίξετε με τον κώδικα και να κάνετε assemble εκ νέου, φροντίστε να συμπεριλάβετε το αρχείο που αντιστοιχεί στο τσιπάκι σας. Αυτό γίνεται από τις γραμμές 54 και 55 του "κεντρικού" αρχείου (vga text-console.asm):

```
.include "m88def.inc"
;.include "m328def.inc"
```

Το ελληνικό ερωτηματικό σηματοδοτεί την αρχή ενός σχολίου. Επομένως, αρκεί να το προσθέσετε στη γραμμή που θέλετε να απενεργοποιήσετε και να το αφαιρέσετε από την άλλη.

Για τον προγραμματισμό του τσιπ χρησιμοποιήσαμε το εργαλείο avrdude. Με το ίδιο πρόγραμμα πετύχαμε και την κατάλληλη ρύθμιση των fuse bytes του τσιπ. Για την πραγματοποίηση των ρυθμίσεων χρησιμοποιήσαμε την εφαρμογή που βρίσκεται στη διεύθυνση www.engbedded.com/fusecalc. Από εκεί, αφού επιλέξαμε το τσιπάκι μας, φροντίσαμε να απενεργοποιήσουμε την εσωτερική διαίρεση της συχνότητας (Divide clock by 8 internally) και να επιλέξουμε τη χρήση εξωτερικού ρολογιού (Ext. Clock). Στη συνέχεια αντιγράψαμε το output που παρήγαγε η εφαρμογή και το χρησιμοποιήσαμε για τη ρύθμιση του τσιπ. Έτσι, καταλήξαμε να εκτελέσουμε κάτι τέτοιο:

```
> avrdude -P COM3 -p m88 -b 19200 -c avrisp -U lfuse:w:0xe0:m -U
hfuse:w:0xdf:m -U efuse:w:0xff:m
```

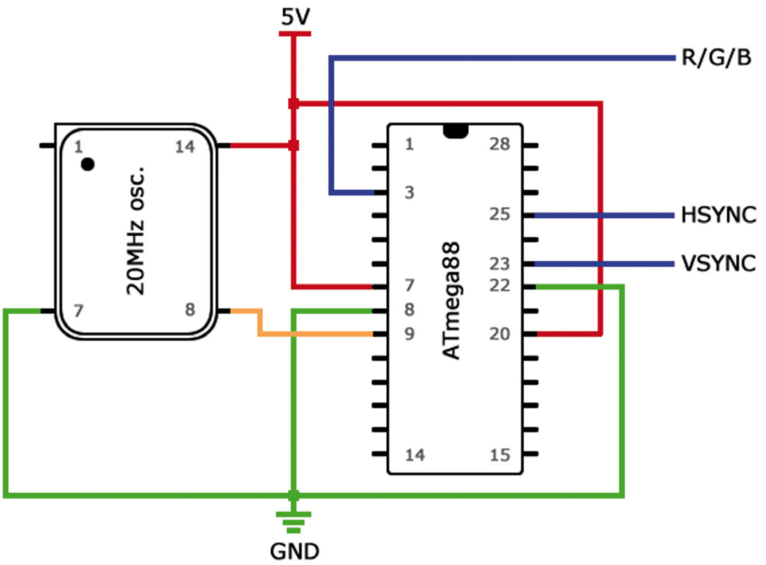
Για να κατανοήσετε καλύτερα τι σημαίνουν τα παραπάνω και να μάθετε πώς μπορείτε να μετατρέψετε το Arduino σε hardware-programmer για τους AVR, δείτε το άρθρο "Απόδραση από το Arduino" (τεύχος 025).

Κύκλωμα

Όπως αναφέραμε και στην αρχή, το κύκλωμά μας είναι εξαιρετικά απλό. Ουσιαστικά αποτελείται από μόλις πέντε εξαρτήματα: Ένα θηλυκό βύσμα VGA, έναν ταλαντωτή (oscillator) στα 20MHz, τον μικροελεγκτή και δύο κεραμικούς πυκνωτές στα 100nF. Προφανώς, το βύσμα απαιτείται για τη σύνδεση κάποιου καλωδίου, που θα οδηγεί το σήμα από τον μικροελεγκτή στην οθόνη. Αναρωτιέστε τώρα ποιες συνδέσεις θα χρειαστεί να γίνουν; Στην εικόνα 5 φαίνεται η επίσημη αρίθμηση των επαφών ενός θηλυκού βύσματος VGA. Οι επαφές 1, 2 και 3 μεταφέρουν την πληροφορία για το κόκκινο, το πράσινο και το μπλε χρώμα, αντίστοιχα. Η επαφή 13 μεταφέρει τους παλμούς HSYNC, ενώ η επαφή 14 μεταφέρει τους παλμούς VSYNC. Τέλος, οι επαφές 5, 6, 7, 8 και 10 συνδέονται στο GND της εκάστοτε οθόνης. Όταν φτάσουμε στο κύκλωμα θα σας πούμε ακριβώς πού πρέπει να συνδεθεί το καθετί.

Ο ταλαντωτής αποτελεί ουσιαστικά ένα ολόκληρο κύκλωμα, πακτωμένο σε ένα μικρό μεταλλικό περίβλημα. Η αποστολή των ταλαντωτών είναι αρκετά απλή: Παράγουν έναν τετραγωνικό παλμό σε μια συγκεκριμένη συχνότητα. Εμείς χρειαζόμαστε έναν ταλαντωτή στα 20MHz, που θα λειτουργεί σαν το "ρολόι" για τον μικροελεγκτή μας. Σ’ αυτό το σημείο μπορεί να σκεφτείτε ότι ένας απλός κρύσταλλος στα 20MHz θα ήταν επίσης αρκετός. Γιατί να καταφύγουμε σε ένα ξεχωριστό κύκλωμα; Ο λόγος είναι πολύ απλός: Οι ταλαντωτές παρουσιάζουν εξαιρετικά μεγάλη σταθερότητα, η οποία υπερβαίνει εκείνη των κρυστάλλων. Και όπως αντιλαμβάνεστε, η σταθερότητα στο ρολόι του επεξεργαστή είναι κρίσιμη, για μια εφαρμογή που απαιτεί ακρίβεια στην κλίμακα των clock cycles.

Οι συνδέσεις που απαιτούνται για το κύκλωμα φαίνονται στην εικόνα 6. Οι επαφές του βύσματος VGA που μεταφέρουν τη χρωματική πληροφορία (R, G και B) συνδέο-



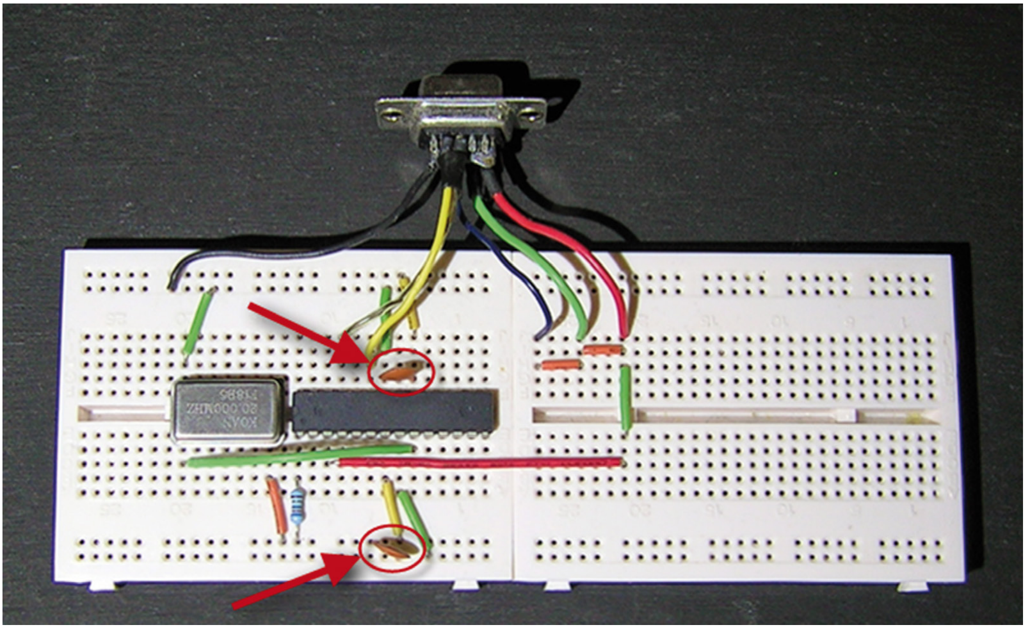
Εικόνα 6
Αν δεν κάνουμε λάθος, πρόκειται για το απλούστερο κύκλωμα που έχουμε δει μέχρι τώρα. Πέρα από τον μικροελεγκτή και τον ταλαντωτή που φαίνονται στο διάγραμμα, θα χρειαστείτε και δύο κεραμικούς πυκνωτές των 100nF. Οι πυκνωτές πρέπει να συνδεθούν μεταξύ GND και 5V, όσο πιο κοντά γίνεται στους ακροδέκτες τροφοδοσίας του μικροελεγκτή!

νται όλες μαζί και πάνω στον ακροδέκτη 3 του μικροελεγκτή. Με αυτή τη σύνδεση, τα pixels θα είναι πάντοτε λευκά (όλα τα χρώματα κορεσμένα) ή μαύρα. Ο ακροδέκτης 3 αποτελεί την έξοδο του συστήματος UART του μικροελεγκτή. Επιλέξαμε να χρησιμοποιήσουμε το συγκεκριμένο σύστημα σαν έξοδο για τα δεδομένα των pixels, επειδή επιτρέπει την αυτόματη σειραϊκή "εκπομπή" ενός ολόκληρου byte. Το πρόγραμμά μας, δηλαδή, ετοιμάζει τα δεδομένα για 8 pixels (8bit, εφόσον έχουμε μονόχρωμη εικόνα), τα παρέχει στο σύστημα UART και μέχρι να ολοκληρωθεί η εκπομπή τους, ετοιμάζει την επόμενη οκτάδα. Με αυτό το τέχνασμα καταφέρνουμε να ικανοποιήσουμε τον (πολύ ζόρικο) περιορισμό των 2 clock cycles ανά pixel. Οι υπόλοιπες συνδέσεις είναι μάλλον κατανοητές. Από το διάγραμμα απουσιάζουν οι δύο κεραμικοί πυκνωτές των 100nF, που πρέπει να συνδεθούν μεταξύ GND και 5V. Κάτι ακόμα που δεν φαίνεται, είναι η σύνδεση των επαφών GND του βύσματος VGA με το GND του κυκλώματος. Είναι κρίσιμο και δεν πρέπει να το ξεχάσετε!

Στην εικόνα 7 φαίνεται η υλοποίηση του κυκλώματος σε ένα breadboard. Παρατηρείστε ότι οι δύο πυκνωτές έχουν τοποθετηθεί όσο πιο κοντά γινόταν στους ακροδέκτες τροφοδοσίας του ATmega88. Αυτή η θέση για τους πυκνωτές είναι επιβεβλημένη, προκειμένου να περιορίσουμε στο μέγιστο βαθμό το θόρυβο της τροφοδοσίας. Ο θόρυβος στην τροφοδοσία δεν αποτελεί συχνό φαινόμενο, ούτε και τόσο μεγάλο πρόβλημα. Στο συγκεκριμένο κύκλωμα, όμως, ο ταλαντωτής παράγει μια σχετικά μεγάλη συχνότητα (20MHz), η οποία μπορεί να διαχυθεί εύκολα σε ένα breadboard με τη μορφή θορύβου.

Διάρθρωση και τεχνάσματα

Το σήμα VGA παράγεται από την ρουτίνα εξυπηρέτησης ενός interrupt (ISR). Το συγκεκριμένο interrupt παράγεται με τη βοήθεια ενός timer, κάθε 636 clock cycles. Αυτό το νούμερο δεν επιλέχθηκε τυχαία. Προέκυψε από το άθροισμα της διάρκειας



Εικόνα 7
Το κύκλωμά μας σε breadboard. Τα βέλη σημειώνουν τους δύο κεραμικούς πυκνωτές, που χρησιμοποιούνται για το φιλτράρισμα της τροφοδοσίας και τον περιορισμό του θορύβου. Έχετε υπόψη σας ότι τα 20MHz είναι πολλά για το «θορυβώδες» περιβάλλον ενός breadboard!

του front porch, του back porch, του horizontal sync pulse και του ορατού τμήματος κάθε γραμμής. Όπως καταλαβαίνετε, κάθε φορά που εκτελείται η ρουτίνα εξυπηρέτησης του interrupt, το πρόγραμμα "εκπέμπει" έναν παλμό οριζόντιου συγχρονισμού καθώς και τα δεδομένα για τα ορατά pixels που απαρτίζουν μια γραμμή (scanline). Ανά 525 γραμμές —όσο διαρκεί ένα πλήρες κάδρο, δηλαδή— ο κώδικας στέλνει κι έναν παλμό κατακόρυφου συγχρονισμού που διαρκεί (παραμένει ενεργός) για δύο γραμμές. Κατά τα άλλα, το πρόγραμμα σχηματίζει μηνύματα κειμένου και τα τοποθετεί σε μια περιοχή της μνήμης SRAM, που βαφτίσαμε "text buffer". Αυτή η περιοχή έχει μέγεθος 768 bytes και χωράει ακριβώς όσους χαρακτήρες μπορεί να προβάλλει ο μικροελεγκτής στην οθόνη (32 χαρακτήρες ανά γραμμή * 24 γραμμές). Κάθε θέση σε αυτή την περιοχή της μνήμης αντιστοιχεί σε μια θέση επί της οθόνης. Η ρουτίνα που παράγει τα scanlines διαβάζει κάθε φορά το κατάλληλο τμήμα του "text buffer", μετατρέπει τους χαρακτήρες σε pixels κι "εκπέμπει" αυτά που πρέπει να προβληθούν στην εκάστοτε γραμμή (scanline).

Τελικά, το πρόγραμμα δεν διατηρεί πουθενά κάποιο "frame buffer". Τα δεδομένα για κάθε pixel παράγονται δυναμικά και δεν υπάρχουν κάπου συγκεντρωμένα και υπολογισμένα από πριν. Αυτή η μέθοδος παραγωγής του σήματος ονομάζεται "racing the beam". Το πρόγραμμα υπολογίζει την τιμή κάθε pixel, λίγα μικροδευτερόλεπτα πριν εμφανιστεί στην οθόνη! Στη συνέχεια θα εξετάσουμε μερικά σημεία του προγράμματος που παρουσιάζουν ιδιαίτερο ενδιαφέρον.

- **Σχηματισμός κάδρου.** Όπως έχουμε αναφέρει, κάθε κάδρο απαρτίζεται από 525 γραμμές (scanlines). Από αυτές, μόνο οι 480 είναι "ενεργές", με την έννοια



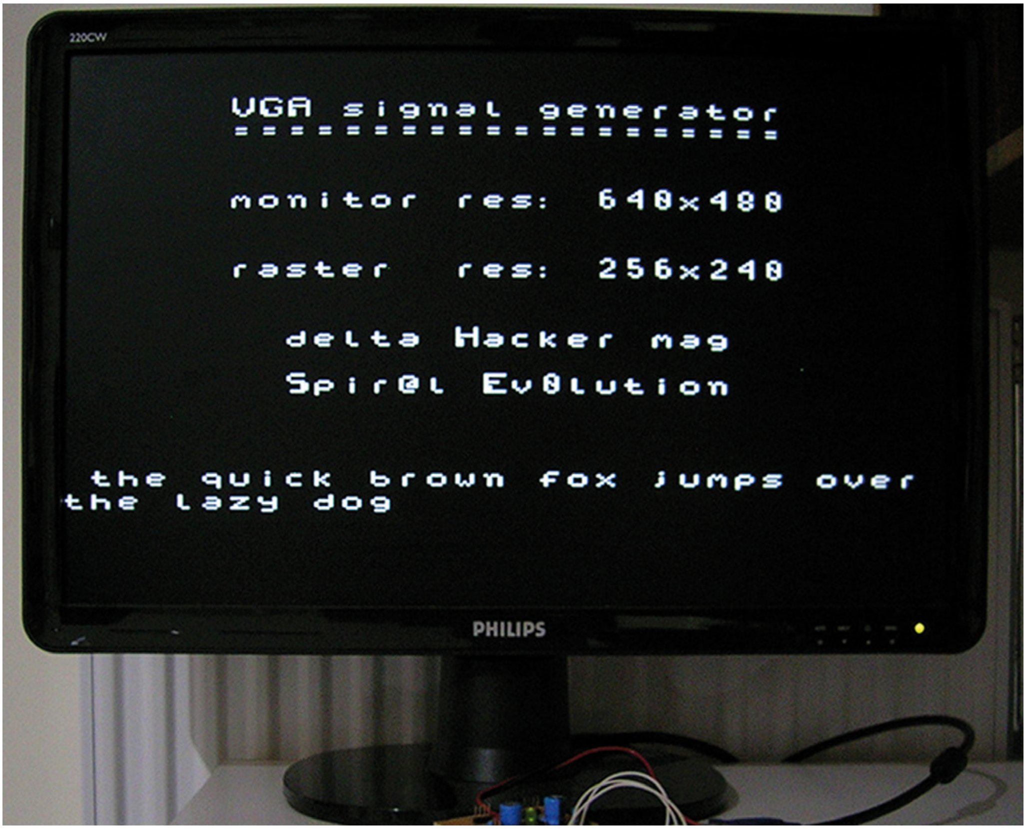
Εικόνα 8
Επιτυχία! Το text buffer δεν αρχικοποιήθηκε και περιλαμβάνει πολλά σκουπίδια, ενώ και ο πίνακας που ορίζει το font δεν έχει ολοκληρωθεί. Ως συνέπεια, εκτός από τους τυχαίους χαρακτήρες, εμφανίζονται και πολλά φωτεινά μπλοκ (χαρακτήρες των οποίων η μορφή δεν έχει οριστεί).

ότι μόνο αυτές προβάλλονται στην οθόνη. Ωστόσο, όλες οι γραμμές (ορατές κι αόρατες) ξεκινούν με έναν παλμό οριζόντιου συγχρονισμού. Η ρουτίνα που σχηματίζει τις γραμμές, λοιπόν, πρέπει να "εκπέμψει" αρχικά έναν παλμό HSYNC και στη συνέχεια είτε να μεταδώσει τα προς εμφάνιση δεδομένα, είτε να τερματιστεί. Με αυτόν τον τρόπο, το υπόλοιπο πρόγραμμα κερδίζει χρόνο για να προχωρήσει τις άλλες εργασίες (συμπλήρωση κειμένου στο text buffer, διαγραφή κειμένου, κ.λπ). Από αυτά συνεπάγεται ότι η ρουτίνα σχηματισμού των γραμμών πρέπει να γνωρίσει σε ποια γραμμή του κάδρου βρίσκεται καθώς και το αν αυτή είναι ορατή ή όχι. Για να πετύχουμε κάτι τέτοιο θα μπορούσαμε να χρησιμοποιήσουμε έναν μετρητή, να τσεκάρουμε την τιμή του και κάθε φορά που φτάνει στο 525 να τον μηδενίζουμε. Αυτή η λύση όμως είναι μπελαλίδικη. Εξάλλου, με δεδομένο ότι ο επεξεργαστής μας είναι 8μπιτος, οι πράξεις με μεγέθη των 16 bits (τόσα απαιτούνται για αριθμούς μεγαλύτερους του 255) εισάγουν καθυστερήσεις που δεν επιτρέπονται στο συγκεκριμένο πρόγραμμα. Η λύση σε αυτό το ζήτημα δίνεται από έναν πίνακα, ο οποίος περιλαμβάνει δύο bytes για κάθε scanline. Η ρουτίνα διαβάζει αυτά τα bytes και αντλεί διάφορες πληροφορίες (το αν η αντίστοιχη γραμμή είναι ορατή ή όχι, ποια γραμμή κει-

μένου εμφανίζεται στη συγκεκριμένη περιοχή της οθόνης και το αν πρέπει να δημιουργηθεί παλμός VSYNC). Κάθε φορά που εκτελείται η ρουτίνα ο δείκτης που διαβάζει τον συγκεκριμένο πίνακα αυξάνεται, ώστε να διαβάσει τα επόμενα bytes. Στο τέλος του πίνακα έχουμε τοποθετήσει ένα byte με κατάλληλη τιμή, που σηματοδοτεί τη συμπλήρωση ενός κάδρου. Όταν η ρουτίνα συναντήσει το συγκεκριμένο byte, γνωρίζει ότι ο πίνακας εξαντλήθηκε και ξεκινά την ανάγνωση από την αρχή. Με άλλα λόγια, το συγκεκριμένο byte μαρκάρει το τέλος του πίνακα και γι' αυτό το λόγο ονομάζεται και "sentry element".

- **Διόρθωση χρονισμού.** Ένα λάθος που κάνουν όλοι όσοι ασχολούνται για πρώτη φορά με τα interrupts, είναι να πιστεύουν ότι η αντίστοιχη ρουτίνα εξυπηρέτησης εκτελείται ακαριαία — ή τέλος πάντων μετά από ένα μικρό αλλά σταθερό χρονικό διάστημα. Στην πραγματικότητα δεν συμβαίνει τίποτα από τα δύο. Από την ενεργοποίηση ενός interrupt και μέχρι την εκτέλεση της αντίστοιχης ρουτίνας εξυπηρέτησης μεσολαβεί ένα χρονικό διάστημα που ονομάζεται "interrupt response time" και το οποίο δεν είναι σταθερό! Βλέπετε, ο μικροελεγκτής δεν μπορεί να διακόψει μια εντολή στη μέση. Επομένως, όταν ενεργοποιείται κάποιο interrupt, ο μικροελεγκτής ολοκληρώνει την τρέχουσα εντολή και στη συνέχεια ξεκινά την εκτέλεση της ρουτίνας εξυπηρέτησης. Όπως θα έχετε υπόψη σας, όμως, οι εντολές δεν απαιτούν όλες τον ίδιο χρόνο εκτέλεσης. Υπάρχουν πολλές που ολοκληρώνονται σε ένα clock cycle, αλλά υπάρχουν κι αρκετές που απαιτούν δύο ή και τρία clock cycles[7]. Αναρωτιέστε τι μας έπιασε και το ρίξαμε στη θεωρία; Εξαιτίας όσων περιγράψαμε, η ρουτίνα που δημιουργεί τους παλμούς συγχρονισμού και "εκπέμπει" τα δεδομένα για τα ορατά pixels, δεν εκτελείται με την απαιτούμενη (και πολύ αυστηρή) περιοδικότητα. Μερικές φορές, η εκτέλεσή της ενδέχεται να καθυστερήσει κατά δύο ή τρία clock cycles. Ε, λοιπόν, αυτή η καθυστέρηση είναι αρκετή για να αλλοιωθεί το σήμα σε τέτοιο βαθμό, ώστε η οθόνη να μην "κλειδώνει" και να μη δείχνει τίποτα! Γι' αυτό το λόγο, στην αρχή της ρουτίνας και πριν ξεκινήσει οποιαδήποτε κρίσιμη εργασία, τοποθετήσαμε μια μικρή καθυστέρηση. Αυτή η καθυστέρηση είναι μεταβλητή και, συγκεκριμένα, είναι αντιστρόφως ανάλογη του χρόνου που μεσολάβησε από την ενεργοποίηση του interrupt ως την κλήση της ρουτίνας. Έτσι, η εκτέλεση των κρίσιμων εργασιών εντός της ρουτίνας ξεκινά πάντοτε στο σωστό χρόνο.
- **Ταχεία αποθήκευση.** Μέσα στη ρουτίνα εξυπηρέτησης του interrupt χρησιμοποιούνται οι καταχωρητές Z και Y[8]. Οι ίδιοι καταχωρητές, όμως, αξιοποιούνται και στο υπόλοιπο πρόγραμμα. Επομένως, στο ξεκίνημα της ρουτίνας και πριν χρησιμοποιηθούν, πρέπει να αποθηκευτούν. Αντίστοιχα, λίγο πριν τον τερματισμό της ρουτίνας, πρέπει να ανακτήσουν τις αρχικές τους τιμές. Το ίδιο ισχύει και για τον καταχωρητή SREG (Status Register), που συμμετέχει ενεργά στην εκτέλεση διαφόρων εντολών (πρόσθεση με κρατούμενο, διακλάδωση υπό συνθήκες κ.λπ). Ο χώρος που αξιοποιείται συνήθως για την προσωρινή αποθήκευση και την ανάκτηση είναι η στοίβα (stack) του συστήματος. Στη ρουτίνα που εξετάζουμε, όμως, τα χρονικά περιθώρια είναι πολύ στενά και δεν επιτρέπουν αναφορές σε αυτήν τη στοίβα. Οι εντολές push και pop, που προσθέτουν ή αφαιρούν στοιχεία, εισάγουν σχετικά μεγάλες καθυστερήσεις. Στο πρόγραμμά μας, λοιπόν, χρησιμοποιήσαμε σαν προσωρινό αποθηκευτικό χώρο μερικούς από τους καταχωρητές του πυρήνα. Μάλιστα, για την μετακίνηση

7. Τουλάχιστον αυτό ισχύει στην περίπτωση των μικροελεγκτών AVR.
8. Πρόκειται για καταχωρητές που χρησιμοποιούνται σαν δείκτες, για την προσπέλαση της μνήμης Flash (Z) και SRAM (Y).



Εικόνα 9
Η εικόνα μιλάει από μόνη της ;)

τιμών από και προς αυτούς, αξιοποιούμε την εντολή movw. Η συγκεκριμένη εντολή έχει τη δυνατότητα να μεταφέρει ζεύγη καταχωρητών (16bit) σε ένα μόνο clock cycle!

Επιπεδα αφαίρεσης

Η ρουτίνα που έχουμε αναφέρει πολλές φορές μέχρι τώρα, κάνει όλη τη βρώμικη δουλειά. Διαβάζει την κατάλληλη γραμμή κειμένου από το text buffer και για κάθε χαρακτήρα που συναντά, κάνει τα εξής: Ανατρέχει σε έναν πίνακα όπου κάθε χαρακτήρας αντιστοιχείται σε ένα σύνολο pixels. Ουσιαστικά, αυτός ο πίνακας ορίζει τη γραμματοσειρά (font) του συστήματός μας. Από αυτόν τον πίνακα διαβάζει την κατάλληλη γραμμή από pixels και την εκπέμπει προς την οθόνη. Αν αυτή η περιγραφή σας μπέρδεψε, μην ανησυχείτε. Θα ήταν εντυπωσιακό αν κατανοούσαμε τα πάντα με την πρώτη ανάγνωση και κυρίως, την πρώτη φορά που ερχόμαστε σε επαφή με το θέμα. Όπως και να ‛χει, η εν λόγω ρουτίνα ολοκληρώνει ένα σύνθετο πλέγμα εργασιών και μάλιστα με τεράστια ακρίβεια! Έτσι, έξω απ’ αυτήν τα πάντα είναι απλούστατα. Το υπόλοιπο πρόγραμμα ασχολείται μόνο με την εισαγωγή μηνυμάτων στο text buffer, με τον καθαρισμό του και γενικότερα με σχετικά απλοϊκές εργασίες. Αν θέλετε να παίξετε κι εσείς με τον κώδικά μας, μπορείτε να προσθέσετε

τον κώδικά σας αμέσως μετά την ετικέτα main. Εκεί μπορείτε να χρησιμοποιήσετε τις βοηθητικές ρουτίνες που ετοιμάσαμε, για να διευκολύνουμε την ανάπτυξη των δικών σας εφαρμογών:

- **Clear.** Καλώντας τη ρουτίνα clear (rcall clear) διαγράφονται όλα τα περιεχόμενα του text buffer και η οθόνη καθαρίζει.
- **Print.** Αυτή η ρουτίνα τυπώνει μηνύματα που βρίσκονται ήδη αποθηκευμένα στη μνήμη FLASH του μικροελεγκτή. Για τη λειτουργία της απαιτεί δύο παραμέτρους: Η πρώτη αποτελεί έναν δείκτη, που δηλώνει από ποιον χαρακτήρα ξεκινά το μήνυμα που θέλουμε να εκτυπωθεί. Βλέπετε, όλα τα μηνύματα βρίσκονται αποθηκευμένα σαν ένα συνεχόμενο string, στο τέλος του προγράμματος και αμέσως μετά την ετικέτα messages. Εκεί μπορείτε να προσθέσετε και τα δικά σας! Η δεύτερη παράμετρος δηλώνει το πλήθος των χαρακτήρων του προς εκτύπωση μηνύματος. Η κλήση της ρουτίνας πραγματοποιείται με τον τυπικό τρόπο που καλούνται ρουτίνες στην Assembly (rcall print). Οι δύο παράμετροι, όμως, πρέπει να έχουν φορτωθεί στους καταχωρητές tmp1 και tmp2.
- **Locate.** Αναρωτιέστε πού θα τυπωθεί το εκάστοτε μήνυμα με μια κλήση της ρουτίνας print; Μα, ακριβώς εκεί όπου βρίσκεται ο (νοητός) δρομέας του συστήματος! Για να μετατοπίσουμε το δρομέα σε συγκεκριμένη γραμμή και στήλη της οθόνης, αρκεί να χρησιμοποιήσουμε τη ρουτίνα locate. Η εν λόγω ρουτίνα λειτουργεί όπως η ομώνυμη εντολή της BASIC: Δέχεται δύο παραμέτρους και η πρώτη δηλώνει την επιθυμητή γραμμή, ενώ η δεύτερη τη στήλη. Όπως και στην περίπτωση της print, οι δύο παράμετροι πρέπει να φορτωθούν στους καταχωρητές tmp1 και tmp2, πριν από την κλήση της ρουτίνας (rcall locate).
- **String.** Αυτή η ρουτίνα είναι επίσης εμπνευσμένη από την BASIC! Με τη βοήθειά της μπορούμε να τυπώσουμε ένα string, το οποίο απαρτίζεται από κ στο πλήθος επαναλήψεις του λ χαρακτήρα. Τα μεγέθη κ και λ (ένας αριθμός κι ένας χαρακτήρας) αποτελούν τις δύο παραμέτρους που δέχεται η συγκεκριμένη ρουτίνα. Οι παράμετροι φορτώνονται και πάλι στους γνωστούς καταχωρητές, πριν από την κλήση της ρουτίνας (rcall string).
- **Delay.** Προφανώς, αυτή η ρουτίνα δεν τυπώνει τίποτα. Η μοναδική παράμετρος που δέχεται δηλώνει τη διάρκεια της επιθυμητής παύσης του προγράμματος, εκφρασμένη σε εκατοντάδες χιλιοστών του δευτερολέπτου (milliseconds). Με άλλα λόγια, αν ορίσουμε την τιμή της παραμέτρου σε κ, η συνολική παύση θα ισούται με κ*100ms.

Θέλουμε να πιστεύουμε ότι η εκτύπωση μηνυμάτων στην οθόνη, με τη βοήθεια ενός μόνο μικροελεγκτή, σας εντυπωσιάζει όσο κι εμάς. Ευελπιστούμε επίσης ότι θα παίξετε με τον κώδικά μας κι ότι θα αναπτύξετε γύρω του τις δικές σας εφαρμογές. Σε κάθε περίπτωση, ελπίζουμε το συγκεκριμένο project να σας παρακινήσει ν' ασχοληθείτε κι εσείς με την αρχαϊκή, στριφνή αλλά και πανίσχυρη Assembly! Κι αν το μονόχρωμο κείμενο σας φαίνεται ανιαρό, μπορείτε να περιμένετε το επόμενο άρθρο, στο οποίο θα προσθέσουμε γραφικά και χρώμα.



Τα επεισόδια του deltaCast, μοιάζουν με ταινίες.
Στη διάρκεια -- όχι στην ποιότητα.
deltaCast s01e11
MetaSentials!
http://deltahacker.gr/category/deltacast

Skill: Beginner
Tags: markdown, markup, plain text, HTML

MARK DOWN

by multiPetros

Απλό κείμενο, με άψογη μορφοποίηση!

Η γλώσσα επισήμανσης Markdown επιτρέπει τη συγγραφή αρχείων απλού κειμένου σε σχεδόν φυσική γλώσσα, ενώ φροντίζει για την εύκολη μετατροπή τους σε έγγραφα HTML. Αν δεν χρησιμοποιείτε ήδη τη Markdown, μετά από αυτό το άρθρο θα απορείτε γιατί ;)

Όλοι έχουμε συναντήσει αρχεία κειμένου με όνομα README.txt, READ.ME ή κάτι παρόμοιο. Τέτοια αρχεία κυκλοφορούν μαζί με όλα σχεδόν τα προγράμματα, τις βιβλιοθήκες και το ψηφιακό υλικό κάθε είδους. Η συντριπτική πλειονότητα αυτών των αρχείων περιλαμβάνει απλό κείμενο (plain text) κι ως εκ τούτου, για την ανάγνωσή τους αρκεί η χρήση ενός απλούστατου text editor[1]. Όσο κι αν έχουν εξελιχθεί οι δυνατότητες και οι ευκολίες στην επεξεργασία κειμένου, οι προγραμματιστές επιμένουν στη χρήση απλού κειμένου, για τη συγγραφή κάθε συνοδευτικού κειμένου. Αυτό δεν συμβαίνει ούτε από βαρεμάρα, ούτε από ιδιοτροπία. Οι λόγοι είναι καθαρά πρακτικοί: Τα αρχεία απλού κείμενου μπορούν να διαβαστούν από οποιοδήποτε σύστημα του παρόντος, του παρελθόντος αλλά και του μέλλοντος. Αυτό εξασφαλίζεται από το γεγονός ότι η εμφάνιση των πληροφοριών δεν απαιτεί λογισμικό που να πραγματοποιεί κάποιου είδους αποκωδικοποίηση. Τα δεδομένα, δηλαδή, είναι άμεσα αναγνώσιμα από τον άνθρωπο.

Για τον καλλωπισμό των αρχείων απλού κειμένου χρησιμοποιούνται συνήθως τα σημεία στίξης και τα σύμβολα. Για παράδειγμα, για την ευκολότερη διάκριση των επικεφαλίδων χρησιμοποιείται συνήθως μια σειρά από επαναλαμβανόμενες παύλες. Μήπως όμως υπάρχει καμία καλύτερη λύση;

Social web για developers

Δεν θα ήταν υπερβολικό αν λέγαμε ότι στις μέρες μας ο προγραμματισμός έχει σ' ένα βαθμό περάσει στο επίπεδο της κοινωνικής δικτύωσης. Βέβαια, θα μπορούσε κανείς να ισχυριστεί και το αντίθετο: Οι τεχνολογίες κοινωνικής δικτύωσης επεκτάθηκαν/εφαρμόστηκαν στο χώρο του προγραμματισμού. Έτσι, υπάρχουν πολλά sites που προσφέρουν φιλοξενία για τον κώδικά μας, υπό τη σκέπη κάποιου version control system. Τα ίδια sites επιτρέπουν τη δημόσια προβολή του κώδικα, ενώ διευκολύνουν και τη συνεργασία με άλλους προγραμματιστές. Δημοφιλές παράδειγμα αποτελεί το GitHub, που στηρίζεται στο σύστημα Git και προσφέρει δωρεάν φιλοξενία για projects Ανοιχτού Λογισμικού, καθώς κι επί πληρωμή λογαριασμούς για προστατευμένη (private) φιλοξενία. Αντίπαλο δέος αποτελεί το Bitbucket, το οποίο παρέχει εντελώς δωρεάν φιλοξενία, δημόσια ή και προστατευμένη, ενώ υποστηρίζει τόσο το σύστημα Mercurial, όσο και το Git. Φυσικά αυτά δεν είναι τα μόνα sites που παρέχουν τέτοιες υπηρεσίες, αλλά είναι σίγουρα τα επιφανέστερα στο χώρο του λεγόμενου social coding.

Μαζί με τον κώδικα, τα εν λόγω sites εμφανίζουν πολλές ακόμα πληροφορίες, για κάθε project. Όλα τα σχετικά δεδομένα αντλούνται αυτόματα από τα αρχεία readme, που βρίσκονται στη ρίζα των καταλόγων κάθε project. Βλέπετε, οι δημιουργοί των sites γνωρίζουν πολύ καλά τη συνήθεια των προγραμματιστών να δημιουργούν και να συντηρούν ένα αρχείο readme για κάθε έργο τους. Βέβαια, η εξαιρετικά λιτή εμφάνιση του απλού κειμένου δεν ταιριάζει ιδιαίτερα στην αισθητική ενός site. Μια πρόχειρη λύση θα ήταν η συγγραφή του αρχείου readme σε δύο μορφές: Σε απλό κείμενο και σε HTML. Όμως κάτι τέτοιο θα απαιτούσε πρόσθετο (και περιττό) κόπο για τον προγραμματιστή, ενώ οι πιθανές ασυνέπειες μεταξύ των δύο εκδοχών θα δημιουργούσαν σύγχυση.

Η γλώσσα Markdown

Ο John Gruber, επηρεασμένος κυρίως από τον τρόπο γραφής των emails απλού κειμένου, δημιούργησε το 2004 τη γλώσσα Markdown. Πρόκειται για μια απλουστευ-

1. Ή η χρήση του cat (άντε, το πολύ και μια διασωλήνωση με το more), για τους απανταχού λάτρεις της κονσόλας.

μένη γλώσσα επισήμανσης (markup language) που έχει σχεδιαστεί κατάλληλα για να μεταφράζεται εύκολα σε HTML. Το μεγάλο προσόν της Markdown, όμως, είναι ότι υιοθετεί τις συμβάσεις που λίγο-πολύ ακολουθούμε όλοι, όταν συντάσσουμε έγγραφα απλού κειμένου. Έτσι, η εκμάθηση της Markdown είναι ιδιαίτερα εύκολη, ενώ τα σχετικά κείμενα είναι ευανάγνωστα από οποιονδήποτε — είτε γνωρίζει τη γλώσσα είτε όχι.

Οι μεγαλύτεροι ιστότοποι στο χώρο του social coding υποστηρίζουν ήδη τη Markdown. Με τη βοήθειά της "μεταφράζουν" τα αρχεία readme σε ευπαρουσίαστα έγγραφα HTML, τα οποία προβάλλουν σαν "αρχική σελίδα" του κάθε project. Εδώ πρέπει να αναφέρουμε ότι, καθώς δεν υπάρχει αυστηρή τυποποίηση για τη γλώσσα, οι διάφοροι ιστότοποι έχουν αναπτύξει τις δικές τους επεκτάσεις στο συντακτικό της. Αυτό πάντως δεν σημαίνει ότι αν μάθουμε Markdown για ένα συγκεκριμένο site, δεν θα μπορούμε να τη χρησιμοποιήσουμε σε κάποιο άλλο ή, τέλος πάντων, στα αρχεία απλού κειμένου που διατηρούμε στον υπολογιστή μας. Πέρα από τις όποιες επεκτάσεις, όλοι οι ιστότοποι κι όλα τα προγράμματα (editors ή και plug-ins) που υποστηρίζουν τη Markdown, αναγνωρίζουν το βασικό συντακτικό που όρισε ο John Gruber. Ας το γνωρίσουμε κι εμείς.

Επικεφαλίδες

Ο κλασικότερος τρόπος επισήμανσης των επικεφαλίδων σ' ένα αρχείο απλού κειμένου, προβλέπει την επαναλαμβανόμενη χρήση ενός συμβόλου. Για παράδειγμα, κάτω από τις κύριες επικεφαλίδες τοποθετούνται γραμμές γεμάτες με το σύμβολο της ισότητας, ενώ κάτω από τις δευτερεύουσες επικεφαλίδες τοποθετούνται γραμμές γεμάτες με το σύμβολο της αφαίρεσης. Ο ίδιος τρόπος επισήμανσης των επικεφαλίδων υιοθετείται και στην Markdown! Μάλιστα, το πλήθος των συμβόλων κάτω από το κείμενο δεν υπόκειται σε περιορισμούς. Ακόμα και αν τα ίσον ή οι παύλες διαφέρουν σε μήκος από τον τίτλο της επικεφαλίδας, η μετατροπή σε HTML θα γίνει ολόσωστα. Βέβαια, για την επισήμανση των επικεφαλίδων στην HTML υπάρχουν αρκετά tags (<h1>, <h2>, <h3> κ.λπ.), καθένα από τα οποία αντιστοιχεί σε διαφορετικό "επίπεδο" επισήμανσης. Γι' αυτό το λόγο, η Markdown υποστηρίζει και έναν ακόμα τρόπο επισήμανσης των επικεφαλίδων, με τη βοήθεια του συμβόλου hash (#). Μπορούμε να προσθέτουμε τον συγκεκριμένο χαρακτήρα από μία έως έξι φορές πριν από το κείμενο της επικεφαλίδας, κι αυτό θα τοποθετείται αργότερα στα αντίστοιχα tags (<h1> έως <h6>). Με μερικά παραδείγματα θα κατανοήσετε αμέσως όσα περιγράψαμε. Εξετάστε για λίγο τις ακόλουθες γραμμές:

```
Επικεφαλίδα πρώτου επιπέδου
===================================
Επικεφαλίδα δεύτερου επιπέδου
---------------------
# Επικεφαλίδα πρώτου επιπέδου
## Επικεφαλίδα δεύτερου επιπέδου
### Επικεφαλίδα τρίτου επιπέδου
...
###### Επικεφαλίδα έκτου επιπέδου
```

Η μετάφρασή τους από Markdown σε HTML, έχει ως εξής:

```
<h1>Επικεφαλίδα πρώτου επιπέδου</h1>
<h2>Επικεφαλίδα δεύτερου επιπέδου</h2>
<h1>Επικεφαλίδα πρώτου επιπέδου</h1>
<h2>Επικεφαλίδα δεύτερου επιπέδου</h2>
<h3>Επικεφαλίδα τρίτου επιπέδου</h3>
...
<h6>Επικεφαλίδα έκτου επιπέδου</h6>
```

Παράγραφοι κι αλλαγές γραμμών

Οι αλλαγές γραμμών στη Markdown δεν επισημαίνονται με κάποιον ειδικό χαρακτήρα, ούτε με κάποιο tag. Για να επιβάλλουμε μια αλλαγή γραμμής, αρκεί να εισαγάγουμε δύο ή περισσότερα κενά και να πατήσουμε ένα [Enter]. Όταν γίνει η μετατροπή σε HTML, τα κενά σε συνδυασμό με την αλλαγή γραμμής (το πάτημα του [Enter]) θα μεταφραστούν σε ένα tag
. Η διάκριση των παραγράφων στην Markdown πραγματοποιείται με μια μόνο κενή γραμμή. Με άλλα λόγια, δεν χρειάζεται να γράψουμε τίποτα για να επισημάνουμε το τέλος μιας παραγράφου. Αντίστοιχα, στο ξεκίνημα μιας νέας παραγράφου δεν χρειάζεται ούτε να πατήσουμε [Tab], ούτε να προσθέσουμε μερικά κενά. Ας δούμε και πάλι μερικά παραδείγματα[2]:

```
Η_απλή_αλλαγή_γραμμής¶
δεν_μεταφράζεται_σε_tag_br¶
¶
Για_να_επιβάλλουμε_την_αλλαγή_γραμμής,_είναι_απαραίτητη_η_χρήση_δύο_
κενών__¶
πριν_από_τον_χαρακτήρα_αλλαγής_γραμμής¶
```

Μετατροπή από Markdown σε HTML:

```
<p>Η απλή αλλαγή γραμμής δεν μεταφράζεται σε tag br</p>
<p>Για να επιβάλλουμε την αλλαγή γραμμής, είναι απαραίτητη η χρήση δύο
κενών<br>
πριν τον χαρακτήρα αλλαγής γραμμής</p>
```

Έμφαση

Για να δώσουμε έμφαση ή να κάνουμε έντονες κάποιες περιοχές του κειμένου μας, πρέπει να τις περικλείσουμε σε αστερίσκους ή σε κάτω παύλες (underscore). Συ-

2. Για να αναδείξουμε τον χαρακτήρα του κενού (SPACE) κάνουμε χρήση του χαρακτήρα της κάτω παύλας (underscore). Για την αλλαγή γραμμής χρησιμοποιούμε το σύμβολο ¶.

γκεκριμένα, αν χρησιμοποιήσουμε έναν αστερίσκο (ή μία κάτω παύλα) στην αρχή και στο τέλος ενός τμήματος κειμένου, η μετάφραση σε HTML θα περικλείσει το συγκεκριμένο απόσπασμα σε ετικέτες <em>. Αν όμως χρησιμοποιήσουμε δύο αστερίσκους (ή δύο κάτω παύλες), η μετάφραση σε HTML θα εισαγάγει το tag <strong>. Οι ακόλουθες δύο γραμμές είναι ισοδύναμες. Παρατηρείστε τις και μετά δείτε τη μετάφρασή τους σε HTML:

```
Διαβάζετε __deltaHacker__, *κάνει καλό*.
Διαβάζετε **deltaHacker**, _κάνει καλό_.
```

Μετάφραση σε HTML:

```
Διαβάζετε <strong>deltaHacker</strong>, <em>κάνει καλό</em>.
Διαβάζετε <strong>deltaHacker</strong>, <em>κάνει καλό</em>.
```

Παράθεση

Όπως θα έχετε προσέξει, όταν ένας email client παραθέτει ένα παλιό μήνυμα μέσα στο σώμα ενός νέου, μαρκάρει το παλιό κείμενο με τον χαρακτήρα "μεγαλύτερο από" (>). Φυσικά, η Markdown υιοθέτησε *και* αυτό το στοιχείο επισήμανσης. Έτσι, όταν μεταφράσουμε ένα κείμενο από Markdown σε HTML, οι γραμμές που ξεκινούν με τον χαρακτήρα ">" τοποθετούνται σε tag <blockquote>. Σημειώστε ότι για την παράθεση μιας ολόκληρης παραγράφου δεν χρειάζεται να προσθέσουμε σε κάθε γραμμή τον χαρακτήρα ">". Αρκεί να τον προσθέσουμε μόνο στο ξεκίνημα της παραγράφου. Τέλος, με την πολλαπλή χρήση του ίδιου χαρακτήρα μπορούμε να πετύχουμε και την εμφωλευμένη παράθεση[3]. Μελετήστε τον ακόλουθο κώδικα σε Markdown:

```
>Το κείμενο αυτό βρίσκεται σε παράθεση
>>που έχει μέσα της άλλη παράθεση
>>>που έχει μέσα της άλλη παράθεση
>Για μία ολόκληρη παράγραφο αρκεί
ένας μόνο χαρακτήρας >
```

Δείτε και την μετάφραση σε HTML:

```
<blockquote>
   <p>Το κείμενο αυτό βρίσκεται σε παράθεση</p>
   <blockquote>
      <p>που έχει μέσα της άλλη παράθεση</p>
      <blockquote>
```

3. Κείμενο σε παράθεση, μέσα σε παράθεση!

```
        <p>που έχει μέσα της άλλη παράθεση</p>
      </blockquote>
    </blockquote>
  </blockquote>
  <blockquote>
  <p> Για μία ολόκληρη παράγραφο αρκεί ένας μόνο χαρακτήρας &gt;</p>
  </blockquote>
```

Λίστες

Για να δημιουργήσουμε μια λίστα με τη βοήθεια της Markdown, αρκεί να τοποθετήσουμε κάθε εγγραφή της λίστας σε ξεχωριστή γραμμή και στην αρχή της να εισαγάγουμε τον χαρακτήρα της αφαίρεσης, της πρόσθεσης ή του πολλαπλασιασμού. Βέβαια, με αυτόν τον τρόπο προκύπτουν μη αριθμημένες λίστες (unordered, όπως λέγονται στην HTML). Για να δημιουργήσουμε μια αριθμημένη λίστα τοποθετούμε πριν από κάθε επιλογή τον αύξοντα αριθμό της κι αμέσως μετά μια τελεία κι ένα κενό. Τώρα, αν θέλουμε να δημιουργήσουμε μια λίστα μέσα σε μία άλλη λίστα, δεν έχουμε παρά να μετατοπίσουμε τις κατάλληλες γραμμές κατά τρεις χαρακτήρες προς τα δεξιά, εισάγοντας ισάριθμα κενά στην αρχή τους. Τέλος, σημειώστε ότι η Markdown δεν θα παραπονεθεί αν βρει στην ίδια λίστα τους χαρακτήρες της πρόσθεσης, της αφαίρεσης και του πολλαπλασιασμού. Ένα δομημένο και καλογραμμένο κείμενο, όμως, θα πρέπει να διέπεται από συνέπεια. Επομένως, σε κάθε λίστα θα ήταν καλό να χρησιμοποιούμε μόνον έναν από αυτούς τους χαρακτήρες. Παρακάτω φαίνεται το κείμενο για μια μη αριθμημένη λίστα, καθώς και για μία αριθμημένη:

```
- Ένα στοιχείο
- Άλλο στοιχείο
- Μια κακογραμμένη λίστα
2. Στοιχείο Α
3. Στοιχείο Β
   1. Στοιχείο Β.1
   2. Στοιχείο Β.2
4. Στοιχείο Γ
5. Στοιχείο Δ
```

Δείτε τον κώδικα που παράγεται από τη μετάφραση της Markdown σε HTML:

```
<ul>
<li>Ένα στοιχείο</li>
<li>Άλλο στοιχείο</li>
```

```
<li>Κάποιο ακόμα στοιχείο</li>
</ul>
<ol>
<li>Στοιχείο A</li>
<li>Στοιχείο B
<ol>
   <li>Στοιχείο B.1</li>
   <li>Στοιχείο B.2</li>
</ol></li>
<li>Στοιχείο Γ</li>
<li>Στοιχείο Δ</li>
</ol>
```

Κώδικας

Ένα πρόγραμμα, σε οποιαδήποτε γλώσσα κι αν είναι γραμμένο, περιλαμβάνει πάντοτε χαρακτήρες που έχουν ειδική ερμηνεία και σημασία. Επιπρόσθετα, σε ένα απόσπασμα κώδικα είναι κρίσιμο να διατηρούμε τη λεγόμενη "οδόντωση" (indentation) και κάθε άλλο στοιχείο μορφοποίησης. Βλέπετε, σε ορισμένες γλώσσες το indentation αποτελεί τμήμα του συντακτικού (π.χ., στην αγαπημένη μας Python), ενώ ταυτόχρονα κάνει τον κώδικα ευανάγνωστο. Επομένως, η μετατροπή σε HTML πρέπει να γίνεται με τέτοιον τρόπο, ώστε η αρχική μορφή του κώδικα, καθώς και όλοι οι χαρακτήρες, να μην αλλοιώνονται καθόλου. Ε, λοιπόν, η γλώσσα Markdown φροντίζει για όλα αυτά. Το μόνο που χρειάζεται να κάνουμε εμείς είναι να βάλουμε τέσσερα κενά ή ένα TAB, πριν από κάθε γραμμή κώδικα. Με τον τρόπο αυτό, κατά τη μετατροπή από Markdown σε HTML οι ειδικοί χαρακτήρες θα αντικατασταθούν από HTML entities κι όλος ο κώδικας θα τοποθετηθεί σε tags <pre> και <code>. Να θυμίσουμε ότι το tag <pre> (από το preformatted) σέβεται και διατηρεί τη μορφή του κειμένου που περικλείει. Ας δούμε την ενσωμάτωση ενός μικρού τμήματος κώδικα[4]:

```
Ο φοβερός μας κώδικας:
____int main() {
______printf("Hello <deltaHacker> world!");
______return 0
____}
```

Τα παραπάνω μεταφράζονται σε HTML ως εξής:

```
<p>Ο φοβερός μας κώδικας:</p>
```

4. Σε αυτό το παράδειγμα χρησιμοποιούμε και πάλι τον χαρακτήρα της κάτω παύλας, για να επισημάνουμε τα κενά.

```
<pre><code>int main() {
   printf("Hello &lt;deltaHacker&gt; world!");
   return 0
}</code></pre>
```

Η γλώσσα Markdown παρέχει λύση και για την περίπτωση που θέλουμε να ενσωματώσουμε κώδικα μέσα σε μια γραμμή κειμένου και όχι σε ξεχωριστό μπλοκ. Σε αυτή την περίπτωση, αρκεί να περικλείσουμε τον κώδικα σε backtick[5]:

```
Ένα script της PHP ξεκινάει πάντα με το `<?php;` και τελειώνει με το
`?>`.
```

Η μετάφραση σε HTML θα δώσει αυτό:

```
Ένα script της PHP ξεκινάει πάντα με το <code>&lt;?php;</code> και
τελειώνει με το <code>?&gt;</code>.
```

Οριζόντιες γραμμές

Ένα έγγραφο περιλαμβάνει συνήθως παραπάνω από μία ενότητες. Ο κλασικός τρόπος διαχωρισμού στα αρχεία απλού κειμένου είναι με τη χρήση οριζόντιων γραμμών, που απαρτίζονται από επαναλαμβανόμενους χαρακτήρες. Για αυτή τη δουλειά η Markdown προβλέπει κάτι παρόμοιο: Μπορούμε να οριοθετήσουμε μια ενότητα τοποθετώντας σε μία κενή γραμμή τρεις αστερίσκους, τρεις κάτω παύλες (underscore) ή τρεις παύλες. Κατά τη μετάφραση σε HTML, αυτές οι γραμμές θα αντικατασταθούν από ένα tag <hr>, το οποίο σχεδιάζει ένα οριζόντιο διαχωριστικό. Εδώ όμως πρέπει να υπογραμμίσουμε μια λεπτομέρεια. Επειδή οι παύλες χρησιμοποιούνται και για την επισήμανση των επικεφαλίδων, αν θέλουμε να ερμηνευτούν σαν μια διαχωριστική γραμμή πρέπει να τοποθετήσουμε ανάμεσά τους από ένα κενό.

Σύνδεσμοι

Οι σύνδεσμοι αποτελούν πιο σύνθετα αντικείμενα σε σχέση με μια επικεφαλίδα ή μία εγγραφή σε κάποια λίστα. Ένας σύνδεσμος στην HTML περιλαμβάνει το target URL και προαιρετικά μια περιγραφή του προορισμού. Στο χρήστη εμφανίζεται η περιγραφή ενώ, όταν αυτή δεν είναι διαθέσιμη, εμφανίζεται η διεύθυνση προορισμού (ένα σκέτο URL). Η Markdown μπορεί να δημιουργήσει συνδέσμους με οποιαδήποτε μορφή και προβλέπει τρεις διαφορετικούς τρόπους επισήμανσης:

- **Απλός σύνδεσμος:** Πρόκειται για έναν σύνδεσμο που εμφανίζει μόνο το target URL (χωρίς περιγραφή). Για τη δημιουργία τέτοιων συνδέσμων με τη Markdown, αρκεί να περικλείσουμε το επιθυμητό URL μέσα στους χαρακτήρες "μικρότερο από" (<) και "μεγαλύτερο από" (>). Ας δούμε ένα παράδειγμα:

5. Πρόκειται για τον χαρακτήρα που βρίσκεται στο ίδιο πλήκτρο με τη λεγόμενη τίλντα (tilde) ή περισπωμένη.

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/
licenses/>.
```

Μετάφραση σε HTML:

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <a href="http://www.gnu.org/
licenses/"> http://www.gnu.org/licenses/</a>.
```

- **Σύνδεσμος με περιγραφή:** Για να δημιουργήσουμε έναν πιο ευπαρουσίαστο σύνδεσμο, μπορούμε να προσθέσουμε μια περιγραφή του προορισμού. Αυτή η περιγραφή πρέπει να τοποθετηθεί μέσα σε "τετράγωνες" αγκύλες ([,]). Αμέσως μετά, πρέπει να συμπληρώσουμε το target URL μέσα σε παρενθέσεις. Ας δούμε και πάλι ένα παράδειγμα:

```
You should have received a copy of the GNU General Public License
along with this program. If not, see [GNU Web Site](http://www.gnu.
org/licenses/).
```

Μετάφραση σε HTML:

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <a href=http://www.gnu.org/
licenses/> GNU Web Site </a>.
```

- **Σύνδεσμος με περιγραφή και αναφορά:** Σε ό,τι αφορά στον παραγόμενο κώδικα HTML, αυτοί οι σύνδεσμοι δεν διαφέρουν σε τίποτα με τους προηγούμενους. Η διαφορά μεταξύ των δύο εντοπίζεται μόνο στον τρόπο επισήμανσης *μέσα* στο έγγραφο του απλού κειμένου. Ένα URL μπορεί να είναι μακροσκελές και να καθιστά δυσανάγνωστο το κείμενο, όταν τοποθετείται στη μέση μιας παραγράφου. Φανταστείτε πόσο παράξενο φαίνεται ένα κείμενο που περιλαμβάνει πέντε-έξι τέτοια URLs. Η λύση που παρέχει η Markdown είναι να μην εισάγουμε το URL μέσα στη ροή του κειμένου, αλλά να βάζουμε στη θέση του μια "ετικέτα αναφοράς". Αυτή την ετικέτα μπορούμε να την ορίσουμε στο τέλος του εγγράφου και να την αντιστοιχίσουμε στο επιθυμητό URL. Οι ετικέτες τοποθετούνται μέσα σε τετράγωνες αγκύλες και χρησιμοποιούνται αμέσως μετά την περιγραφή του συνδέσμου. Για τον ορισμό μιας ετικέτας, αρκεί να τη γράψουμε στην αρχή μιας κενής γραμμής και μέσα σε αγκύλες, να πληκτρολογήσουμε τον χαρακτήρα της "άνω και κάτω τελείας" και στη συνέχεια να δώσουμε το αντίστοιχο URL. Ένα παράδειγμα θα βοηθήσει και πάλι:

```
You should have received a copy of the GNU General Public License
along with this program. If not, see [GNU Web Site][gnu].

...

...

[gnu]: http://www.gnu.org/licenses/
```

Μετάφραση σε HTML:

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <a href=http://www.gnu.org/
licenses/>GNU Web Site</a>.
```

Στο παραπάνω παράδειγμα δημιουργήσαμε την ετικέτα αναφοράς [gnu]. Ο ορισμός της πραγματοποιείται στο τέλος του εγγράφου, όπου και γίνεται η αντιστοίχιση με το επιθυμητό URL. Τέλος, σημειώστε ότι δεν γίνεται καμία διάκριση μεταξύ πεζών και κεφαλαίων σε σχέση με τα ονόματα των ετικετών. Έτσι οι ετικέτες [GNU], [gnu] και [Gnu] θεωρούνται ταυτόσημες.

Εικόνες

Σε αντίθεση με τα έγγραφα HTML, στα αρχεία απλού κειμένου δεν μπορούμε να προβάλλουμε εικόνες. Έχουμε όμως τη δυνατότητα να περιγράψουμε τη θέση μια εικόνας. Με τη βοήθεια της Markdown, μπορούμε να επισημάνουμε με συγκεκριμένο τρόπο αυτή τη θέση, ώστε όταν κάνουμε τη μετάφραση σε HTML να προκύψει ένα κατάλληλο tag <img> και η εικόνα να ενσωματωθεί στο τελικό έγγραφο. Η επισήμανση μιας εικόνας στη Markdown ξεκινά με το χαρακτήρα του θαυμαστικού (!). Στη συνέχεια, γράφουμε μέσα σε τετράγωνες αγκύλες την περιγραφή της εικόνας (το λεγόμενο και alternative text) και, αμέσως μετά, δίνουμε τη θέση της εικόνας μέσα σε παρενθέσεις. Νομίζουμε ότι μετά από όλα τα παραδείγματα που έχουμε εξετάσει, αρκεί να δείξουμε τη γενική μορφή επισήμανσης μιας εικόνας:

```
![image_description](URL_to_image)
```

Μετάφραση σε HTML:

```
<img alt="image_description" src="URL_to_image">
```

Τέλος, όπως και με τους συνδέσμους, έτσι και με τις εικόνες μπορούμε να χρησιμοποιήσουμε ετικέτες αναφοράς, ώστε να μην παρεμβάλλουμε τη διεύθυνσή τους μέσα στη ροή του κειμένου:

```
![image_description][some_id]
```

```
...
...
[some_id]:URL_to_image
```

Μετάφραση σε HTML:

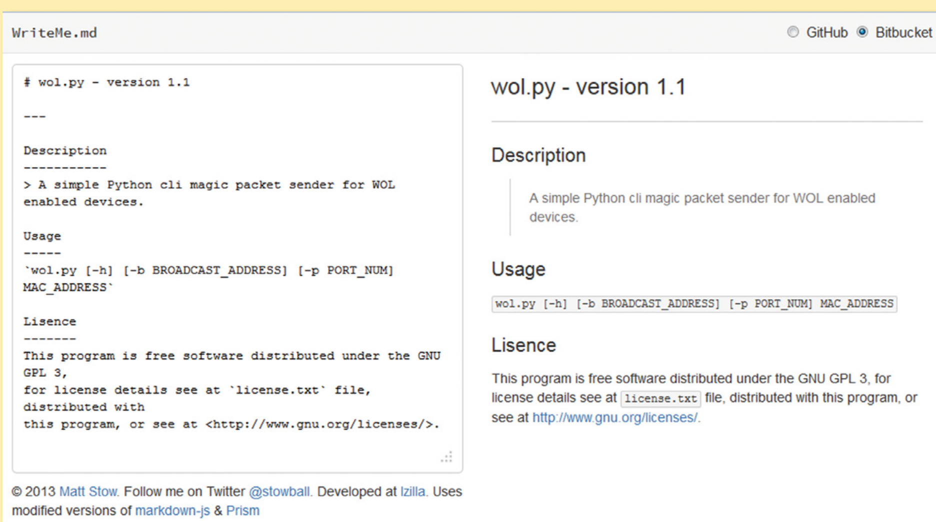
```
<img alt="image_description" src="URL_to_image">
```

Χαρακτήρας διαφυγής

Όπως είδαμε, για την επισήμανση του κειμένου η Markdown επιστρατεύει μερικούς απλούς χαρακτήρες, στους οποίους αποδίδει ξεχωριστή ερμηνεία. Έτσι, για να χρησιμοποιήσουμε τους συγκεκριμένους χαρακτήρες με την κυριολεκτική τους ερμηνεία, πρέπει προηγουμένως να απενεργοποιήσουμε την όποια λειτουργία επιτελούν. Για να πετύχουμε κάτι τέτοιο αρκεί να χρησιμοποιήσουμε τον χαρακτήρα διαφυγής backslash (\). Συγκεκριμένα, αρκεί να τον τοποθετήσουμε ακριβώς πριν από τον χαρακτήρα που θέλουμε να ερμηνευτεί κυριολεκτικά.

Εργαλεία σύνταξης

Για τη σύνταξη αρχείων Markdown δεν απαιτείται ειδικός editor. Κάτι τέτοιο θα ερχόταν σε αντίθεση με το πνεύμα της γλώσσας, όπως και με την όλη φιλοσοφία πίσω από τη γραφή σε απλό κείμενο. Φυσικά, αν εσείς θέλετε να χρησιμοποιήσετε οπωσδήποτε κάποιον editor, έχετε αρκετές επιλογές. Υπάρχουν μερικοί λιγότερο γνωστοί editors που υποστηρίζουν εγγενώς τη γλώσσα Markdown, όπως επίσης και plug-ins

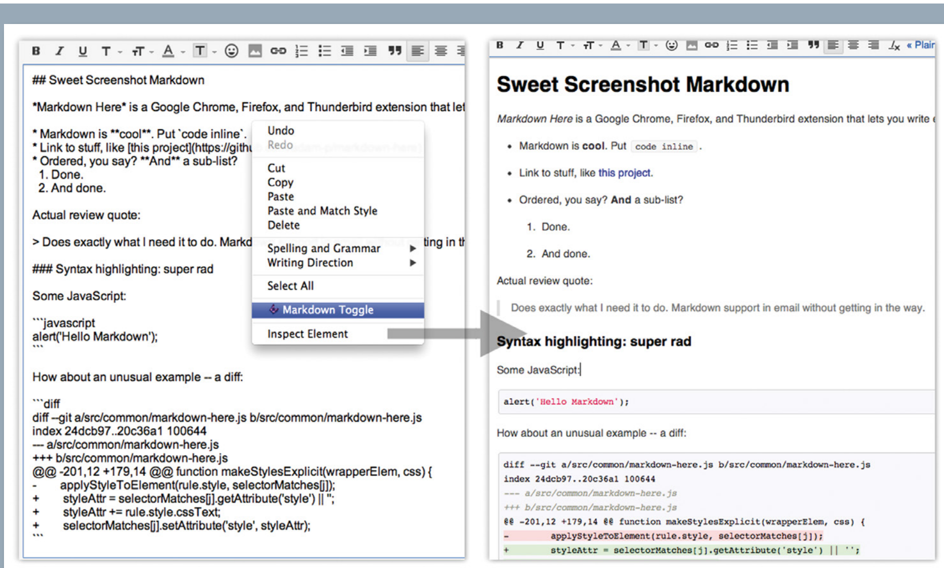


Ο online editor WriteMe.md παρέχει τη δυνατότητα προεπισκόπησης του παραγόμενου κώδικα HTML, με βάση τα style sheets των Github και Bitbucket.

για τους δημοφιλείς. Πάντως μια καλή λύση αποτελούν και οι online editors, οι οποίοι προσφέρουν τη δυνατότητα προεπισκόπησης του παραγόμενου κώδικα HTML! Αξιόλογο παράδειγμα αποτελεί ο WriteMe.md (`http://writeme.mattstow.com`), ο οποίος υποστηρίζει τα style sheets των Github και Bitbucket. Γράφοντας τα κείμενά μας εκεί, μπορούμε να βλέπουμε αμέσως πώς θα εμφανίζονται στα δύο δημοφιλή social coding sites.

Πέρα από τις λύσεις που αναφέραμε, διατίθενται και plug-ins για τους περισσότερους browsers. Άξιο αναφοράς είναι το Markdown Here (`http://markdown-here.com`), που διατίθεται για Chrome, Firefox και Safari. Το συγκεκριμένο plug-in μετατρέπει τον κώδικα Markdown σε κώδικα HTML και συνεργάζεται με τους editors αρκετών δικτυακών υπηρεσιών. Έτσι, με τη βοήθειά του μπορούμε να δημιουργούμε νέες ιστοσελίδες για το Google Sites, το Blogger ή το Wordpress, γράφοντας το περιεχόμενό μας σε κώδικα Markdown και πατώντας μόνο ένα κουμπί! Το Markdown Here συνεργάζεται άψογα και με τον editor του Gmail, που σημαίνει ότι μπορούμε να γράφουμε τα μηνύματά μας σε μορφή απλού κειμένου και να τα μετατρέπουμε εύκολα σε όμορφα έγγραφα HTML.

Η Markdown απαιτεί ελάχιστη προσπάθεια εκμάθησης. Είναι χτισμένη πάνω στον τρόπο με τον οποίο γράφουμε ήδη τα αρχεία απλού κειμένου. Μάλιστα, εξαιτίας της αποδοχής της από τα social coding sites, μπορούμε να πούμε με μια σχετική βεβαιότητα ότι ήρθε για να μείνει. Κάτι τέτοιο βέβαια προϋποθέτει την τυποποίησή της. Η σχετική προσπάθεια δεν έχει ολοκληρωθεί ακόμα, αλλά τουλάχιστον έχει ξεκινήσει. Όπως και να 'χει, εμείς την αγαπήσαμε από την πρώτη στιγμή που τη γνωρίσαμε και τη χρησιμοποιούμε σε όλα μας τα projects. Εσείς;



Το plug-in Markdown Here σε δράση! Μόλις μετέτρεψε σε HTML το κείμενο ενός email, γραμμένο σε Markdown.

Skill: Intermediate
Tags: FreeBSD, virtualization, jails, service jails, complete jails

FreeBSD jails: There is no escape!

Φυλακή: Ένα μέρος που μάλλον δεν έχετε πάει, εκτός ίσως από όταν ήσαστε μικροί και παίζατε Monopoly. Όπως και να 'χει, σίγουρα ούτε τώρα θα θέλετε να πάτε φυλακή και, γενικά, στο άκουσμα και μόνο της λέξης στο μυαλό σάς έρχονται μόνο αρνητικές σκέψεις. Εκτός δηλαδή κι αν είστε χρήστης ή μάλλον διαχειριστής ενός FreeBSD server. Τώρα, θα μου πείτε, τι δουλειά έχει ένα σύστημα FreeBSD με φυλακές; Κοιτάξτε, αν είναι να δώσουμε σ' αυτή τη λέξη μια θετική διάσταση, μόνο με το FreeBSD θα μπορέσουμε να το κάνουμε.



by sonic2000gr

Εκτός αν τα τελευταία χρόνια ζούσατε κάτω από μια πέτρα™, μάλλον έχετε αντιληφθεί τον πολύ θόρυβο που γίνεται γύρω από το virtualization: Χωρίς υπερβολή, ο κόσμος μας έχει γεμίσει εικονικές μηχανές. Δεν θεωρούμε απίθανο ακόμα και στο desktop σας, εσείς, οι ψαγμένοι μας αναγνώστες, να χρησιμοποιείτε το VirtualBox, το VMware ή κάποια άλλη εφαρμογή, αν μη τι άλλο για να έχετε εικονικές μηχανές με διάφορα λειτουργικά-παιχνίδια. Έχετε κάποιο απαραίτητο πρόγραμμα που τρέχει μόνο σε Windows; Κανένα πρόβλημα. Μνήμη να έχετε και δυνατό επεξεργαστή. Θέλετε να δοκιμάσετε τη νέα έκδοση του PC-BSD, για να δείτε αν θα το εγκαταστήσετε στο επόμενό σας desktop; Πανεύκολο: Δύο, τρία κλικ και το μηχάνημά σας τρέχει το PC-BSD μέσα σ' ένα παράθυρο. Δεν σας κάνει; Δεν πειράζει, λίγα μόνο ευχάριστα απογεύματα χρειάζονται και το FreeBSD γίνεται ωραιότατο desktop!

Το virtualization, φίλες και φίλοι, είναι παντού. Ίσως μάλιστα το desktop virtualization να είναι το λιγότερο σημαντικό. Πολλοί από εσάς ίσως έχετε ήδη έναν virtual server, σε κάποιον ISP. Οι εικονικές μηχανές επιτρέπουν πολύ καλύτερη εκμετάλλευση του hardware ενός physical server, ενώ επιτρέπουν την ταυτόχρονη εκτέλεση πολλών λειτουργικών στο ίδιο μηχάνημα. Μερικές φορές σαν όνειρο από το μέλλον ακούγονται όλα αυτά, η ιδέα όμως είναι πάρα πολύ παλιά (διαβάστε το σχετικό άρθρο στο http://everythingvm.com/content/history-virtualization.

Πέρα από το desktop virtualization υπάρχει κι ένας άλλος τρόπος προκειμένου να εκμεταλλευτούμε στο έπακρο το μηχάνημά μας: Φανταστείτε να μπορούσαμε να εκτελούμε διεργασίες με τέτοιον τρόπο, ώστε αυτές να βλέπουν ως root κατάλογο του δίσκου έναν άλλον κατάλογο τον οποίο θα έχουμε εμείς ορίσει. Τι συμφέρον θα είχαμε από μια τέτοια διευθέτηση;

Ας υποθέσουμε ότι είστε διαχειριστής ενός web server. Κατά πάσα πιθανότητα χρησιμοποιείτε τον γνωστό Apache web server. Τον έχετε ρυθμίσει μια χαρά, ώστε να εκτελείται και να εξυπηρετεί διαβάζοντας τα αρχεία του site σας από κάποιο συγκεκριμένο κατάλογο και τους υποκαταλόγους του. Και ξαφνικά, μια ωραία μέρα έρχεται η συμφορά: Ίσως κάποιο δικό σας script δεν ήταν ασφαλές, ίσως οι ρυθμίσεις του ίδιου του server άφηναν περιθώρια για επίθεση, ίσως πάλι δεν είχατε την τελευταία έκδοση του server με τα πλέον πρόσφατα security patches, όπως και να 'χει το θέμα είναι ότι κάποιος malicious user βρήκε τρόπο, μέσα από τη σελίδα σας, να αποκτήσει πρόσβαση στους υπόλοιπους καταλόγους του filesystem. Μέχρι να το καταλάβετε, σας έχουν κλέψει αρχεία με κωδικούς, πιστωτικές κάρτες κι ό,τι άλλο προσωπικό δεδομένο είχατε. Γιατί, εκεί που ο Apache είχε πρόσβαση μόνο σε ένα-δύο καταλόγους, ξαφνικά τον υποχρέωσαν να διαβάσει όλο το filesystem.

Αν όμως είχατε πείσει τον Apache ότι όλο το root filesystem είναι ένας άλλος κατάλογος, η ζημιά θα είχε περιοριστεί εκεί μέσα. Και φυσικά, μέσα στο ψεύτικο αυτό root, δεν θα υπήρχε καμιά άλλη χρήσιμη και ευαίσθητη πληροφορία. Η διαδικασία για να μεταβάλλουμε τον root κατάλογο μιας διεργασίας είναι σχετικά απλή, ενώ υποστηρίζεται τόσο στο Linux όσο και στο FreeBSD με την εντολή.

Η ιδέα του chroot φαίνεται πολύ καλή, αλλά έχει και κάποιους —μάλλον προφανείς— περιορισμούς: περιορίζει *μόνο* την πρόσβαση στο σύστημα αρχείων. Μια διεργασία που τρέχει μέσα στο chroot δεν μπορεί να δει τα αρχεία έξω από τον κατάλογο που έχουμε ορίσει, όμως μπορεί να δει διάφορα άλλα resources του συστήματος. Για παράδειγμα, μπορεί να δει τους χρήστες και τις διεργασίες που εκτελούνται στο σύστημα, ενώ κοινές είναι και οι ρυθμίσεις του δικτύου. Κατά καιρούς μάλιστα έχουν

βρεθεί διάφοροι τρόποι που μπορεί κάποιος έξυπνος επιτιθέμενος να χρησιμοποιήσει, προκειμένου να βγει εκτός των ορίων ενός chroot environment.

Έπειτα, το chroot προσφέρει μεν κάποια αυξημένη ασφάλεια, αλλά πως μπορεί να μας βοηθήσει να εκμεταλλευθούμε το hardware μας, όπως ένα virtual machine; Στα εικονικά μηχανήματα, για παράδειγμα, κάθε guest OS μπορεί να έχει τη δική του εικονική κάρτα δικτύου. Έτσι, κάθε VM μπορεί να τρέχει τον δικό του Apache, στη δική του διεύθυνση δικτύου. Κάτι τέτοιο είναι αδύνατο σε ένα απλό chroot: Το υποσύστημα δικτύωσης παραμένει ένα και είναι κοινό για όλα τα chroot environments. Αλλά μην ανησυχείτε: Ήρθε η ώρα να σας βάλουμε φυλακή!

FreeBSD jails

Αναγνωρίζοντας τα καλά χαρακτηριστικά που μπορεί να προσφέρει το chroot, οι FreeBSD developers το πήγαν αρκετά βήματα παραπέρα. Τι θα λέγατε αν παίρναμε τη βασική φιλοσοφία της απομόνωσης του συστήματος αρχείων και την επεκτείναμε, ούτως ώστε...

- Οι διεργασίες που εκτελούνται μέσα στο chroot να μην έχουν πρόσβαση στις διεργασίες του κανονικού συστήματος
- Οι χρήστες μέσα στο chroot να είναι διαφορετικοί από εκείνους του κανονικού συστήματος και να μην έχουν πρόσβαση στους εξωτερικούς. Ο root χρήστης του chroot δεν έχει κανένα ιδιαίτερο δικαίωμα στο κανονικό σύστημα.
- Το υποσύστημα δικτύωσης να είναι ξεχωριστό και κάθε chroot να έχει δυνατότητα χρήσης διαφορετικού IP, χωρίς όμως το σύστημά μας να χρειάζεται απαραίτητα παραπάνω από μία (φυσική) κάρτα δικτύου.

Όλα αυτά που μόλις αναφέραμε *δεν* τα κάνει το απλό chroot. Καλώς ήλθατε στα FreeBSD jails!

Όπως καταλαβαίνετε, τα FreeBSD jails μάς προσφέρουν τη δυνατότητα να δημιουργούμε, ουσιαστικά, virtual machines μέσα στο FreeBSD σύστημά μας, το οποίο θα το λέμε από εδώ και στο εξής "host". Η υποστήριξη των jails προφανώς ξεκινά από τον ίδιο τον πυρήνα του FreeBSD, ο οποίος μπορεί να διαχωρίζει ποιες διεργασίες ανήκουν στο host και ποιες στα jails. Σε γενικές γραμμές, ένα jail χαρακτηρίζεται από τέσσερα στοιχεία:

1. Μια δομή καταλόγων που αποτελούν το δικό του root filesystem, ακριβώς όπως και στο chroot.
2. Ένα όνομα μηχανήματος (hostname), διαφορετικό από εκείνο του host.
3. Μια δική του διεύθυνση IP. Η διεύθυνση αυτή ορίζεται κατά τη δημιουργία του jail και δεν μπορεί να αλλάξει κατά την εκτέλεσή του. Συνήθως πρόκειται για μια δεύτερη διεύθυνση που δίνουμε στη βασική κάρτα δικτύου του host.
4. Μια "εντολή" που εκτελείται μέσα στο περιβάλλον του jail. Για παράδειγμα, ένα shell στο οποίο δίνουμε άλλες εντολές, ένας server που έχουμε "κλείσει" εκεί μέσα κ.ο.κ.

Και φυσικά, το jail έχει τους δικούς του χρήστες και τον δικό του root, ο οποίος πρέπει να σημειωθεί πως έχει δικαιώματα *μόνο* μέσα σε αυτό. Τώρα, σας είπαμε άραγε ότι μπορείτε να τρέχετε όσα jails θέλετε στο μηχάνημά σας, και μάλιστα ταυτόχρονα;

Complete jails & Service jails

Αν έχετε μπει στο νόημα από την έως τώρα συζήτησή μας, μπορείτε μάλλον να φανταστείτε πώς θα είναι η εσωτερική δομή ενός jail: Μέσα στον κατάλογο που έχουμε ορίσει ως root, θα βρείτε όλους τους γνωστούς καταλόγους ενός πλήρους FreeBSD συστήματος: bin, lib, usr, etc κ.ο.κ. Αν μάλιστα έχετε εγκαταστήσει κι εφαρμογές μέσα στο jail, θα βρείτε και τα γνωστά usr/local και usr/local/etc. Μην περιμένετε βέβαια να χρησιμοποιήσετε το jail για να εκτελέσετε πολλαπλά περιβάλλοντα γραφικών: Σκοπός κάθε jail είναι η εκτέλεση προγραμμάτων κονσόλας και, κατά κύριο λόγο, διαφόρων υπηρεσιών.

Στην περίπτωση που επιθυμούμε να εκτελέσουμε μόνο μια, δυο, συγκεκριμένες εφαρμογές εντός ενός jail, είναι ίσως άσκοπο να αντιγράψουμε όλα τα αρχεία ενός πλήρους FreeBSD base system εκεί μέσα. Για παράδειγμα, ένας Apache web server δεν χρειάζεται όλες τις βιβλιοθήκες και τα shells του FreeBSD για να εκτελεστεί. Θα μπορούσαμε να βρούμε μόνο τα απολύτως απαραίτητα αρχεία και να φτιάξουμε ενα jail με ένα minimum FreeBSD σύστημα, το οποίο θα ήταν σε θέση να εκτελεί μόνο τον Apache και τίποτε άλλο. Όχι μόνο θα κερδίζαμε σε χώρο, αλλά θα αυξάναμε την ασφάλεια ακόμα περισσότερο: είναι δύσκολο να πάρει κάποιος shell access σε ένα σύστημα στο οποίο δεν υπάρχει ούτε ένα shell κι ούτε ένας compiler για να το μεταγλωττίσει!

Ένα τέτοιο jail, το οποίο δεν είναι πλήρες αλλά προορίζεται μόνο για την εκτέλεση συγκεκριμένης εφαρμογής, ονομάζεται **service jail** σε αντίθεση με το **full** ή **complete jail**, το οποίο περιέχει ένα πλήρες FreeBSD base system. Τα service jails είναι μια καλή ιδέα, αλλά η υλοποίησή τους μπορεί να είναι κομματάκι δύσκολη: Δεν είναι πάντα προφανές ποια αρχεία του FreeBSD είναι απαραίτητα σε μια εφαρμογή, προκειμένου αυτή να εκτελείται χωρίς προβλήματα. Η λογική της δημιουργίας ενός service jail έχει ως εξής: Αντιγράφουμε τα βασικά, εκτελούμε την εφαρμογή μας, βλέπουμε τα μηνύματα λάθους του τύπου "δεν βρήκα αυτό το αρχείο", αντιγράφουμε ξανά, δοκιμάζουμε και πάει λέγοντας. Καθώς καταλαβαίνετε, η δημιουργία service jails είναι ένας εκπληκτικός τρόπος να περάσετε πολλά ευχάριστα, χειμωνιάτικα απογεύματα. [Σ.τ.Ε. Αυτά είναι τα καλύτερα, όχι σαν τα καλοκαιρινά :P] Από την άλλη, η δημιουργία ενός πλήρους jail είναι σχετικά απλή υπόθεση και μπορεί να είναι χρήσιμη ακόμη κι αν δεν θέλετε να τρέξετε servers μέσα σε αυτό.

Δημιουργία complete jail

Για να δημιουργήσετε ένα jail χρειάζεται καταρχάς να έχετε τα FreeBSD sources για την έκδοση του FreeBSD που θέλετε να έχει το jail. Ναι: Είναι δυνατόν να έχετε άλλη FreeBSD έκδοση στο host και άλλη στο jail. Μπορείτε ακόμη και να έχετε 64bit host και 32bit jail, αλλά για το δικό μας πείραμα θα εγκαταστήσουμε στο jail την ίδια έκδοση/εκδοχή του FreeBSD με αυτή που έχει το host computer.

Ελπίζουμε να έχετε εγκαταστήσει τα sources κατά την εγκατάσταση του FreeBSD σας. Αν όχι, θα πρέπει να τα κατεβάσετε και να τα αποσυμπιέσετε στον κατάλογο /usr/src του συστήματος. Για παράδειγμα, για το 9.2-RELEASE —κι ως root— δίνουμε:

```
# cd /root
# fetch \
ftp://ftp.otenet.gr/pub/FreeBSD/releases/'uname -m'/9.2-RELEASE/src.txz
```

Αποσυμπιέζουμε το αρχείο src.txz μέσα στο root directory:

```
# cd /
# tar xvzf /root/src.txz
```

Η διαδικασία για την εγκατάσταση του jail περιλαμβάνει ένα πλήρες buildworld, δηλαδή μια μεταγλώττιση όλου του FreeBSD base system από τα sources. Αυτά τα αστεία που ξέρατε με τον Atom και τα Raspberry Pi, λοιπόν, να τα ξεχάσετε! Έτσι, αρχικά, ορίζουμε το root directory για το jail μας. Εμείς επιλέξαμε το /scratch/jailroot, αλλά εσείς βάλτε ό,τι σας βολεύει:

```
# setenv DESTDIR /scratch/jailroot
# mkdir -p $DESTDIR
# cd /usr/src
# make buildworld
```

Αν έχετε μηχάνημα με πολλαπλούς πυρήνες (το καλό που σας θέλουμε, δηλαδή), μπορείτε να επιταχύνετε τη διαδικασία ορίζοντας παράλληλη μεταγλώττιση. Για παράδειγμα, σε μηχάνημα με 4 πυρήνες γράψτε:

```
# make -j5 buildworld
```

(ο γενικός κανόνας είναι -j αριθμός_πυρήνων+1). Αν έχετε έναν πυρήνα και κανένα αργό μηχάνημα, τώρα είναι ώρα να παραγγείλετε εκείνη την πίτσα. Σε περίπτωση που είστε από τους τυχερούς/παράξενους ανθρώπους που έχουν ήδη κάνει buildworld στο σύστημά τους για κάποιον άλλο λόγο, δεν χρειάζεται να το ξανακάνετε. Έχετε ήδη τα αρχεία και μπορείτε να προχωρήσετε στο επόμενο βήμα, που είναι η εγκατάσταση:

```
# make installworld DESTDIR=$DESTDIR
```

Τέλος, θα εγκαταστήσουμε όλα τα αρχεία ρυθμίσεων του /etc που περιλαμβάνονται στο src:

```
# make distribution DESTDIR=$DESTDIR
```

Κάποια απ' αυτά θα χρειαστεί, προφανώς, να τα αλλάξουμε με το χέρι, μετά την εγκατάστασή τους. Στο σημείο αυτό καλό είναι να προσαρτήσουμε και το devfs μέσα στο jail μας:

```
# mount -t devfs devfs $DESTDIR/dev
```

Πριν ξεκινήσουμε το jail, θα πρέπει να κάνουμε κάποιες ρυθμίσεις. Ας αρχίσουμε από το δίκτυο. Το μηχάνημα μας έχει ήδη μια κάρτα δικτύου, ας υποθέσουμε την rl0:

```
# ifconfig
rl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        inet 192.168.0.3 netmask 0xffffff00 broadcast 192.168.0.255
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
```

Θα χρειαστεί να προσθέσουμε μια διεύθυνση δικτύου για να τη χρησιμοποιήσει το jail. Στο παράδειγμά μας, το host βρίσκεται στο 192.168.0.3 και θα βάλουμε το jail στο 192.168.0.10. Προφανώς, δεν είναι απαραίτητο να βρίσκεται καν στο ίδιο subnet, αλλά θα βολέψει στο πείραμά μας. Ας προσθέσουμε ένα alias στο υπάρχον ethernet interface:

```
# ifconfig rl0 192.168.0.10 netmask 255.255.255.0 alias
```

Αν κάνουμε επανεκκίνηση (εννοούμε αν πέσει το ρεύμα, φυσικά — όλοι ξέρουμε ότι δεν χρειάζεται ποτέ να κάνουμε επανεκκίνηση στο FreeBSD), η παραπάνω ρύθμιση θα χαθεί. Χρειάζεται λοιπόν να βάλουμε την ανάλογη γραμμή στο /etc/rc.conf:

```
ifconfig_rl0_alias0="inet 192.168.0.10 netmask 255.255.255.0"
```

Είμαστε έτοιμοι να δηλώσουμε το jail στο rc.conf και να του δώσουμε τις ρυθμίσεις που θέλουμε. Προσθέστε τις παρακάτω γραμμές, αλλάζοντας, όπου χρειάζεται, κατάλληλα τις παραμέτρους:

```
jail_enable="NO"
```

Αν θέλουμε το jail να ξεκινά αυτόματα με την εκκίνηση του συστήματος, βάζουμε "YES". Διαφορετικά, θα το ξεκινάμε χειροκίνητα.

```
jail_list="pkgbuilder"
```

Εδώ βάζουμε μια λίστα με τα ονόματα όλων των jails μας, χωρισμένα με κενά. Βλέπετε ότι το ένα και μοναδικό που φτιάχνουμε, το ονομάζουμε "pkgbuilder".

```
jail_pkgbuilder_rootdir="/scratch/jailroot"
```

Για καθένα από τα jails ορίζουμε το root directory με μια γραμμή του τύπου jail_<jailname>_rootdir="path_to_jail_dir"

```
jail_pkgbuilder_hostname="pkgbuilder"
```

Για κάθενα από τα jails ορίζουμε το hostname, με μια γραμμή της μορφής jail_<jailname>_hostname="a_hostname"

```
jail_pkgbuilder_ip="192.168.0.10"
```

Για καθένα από τα jails ορίζουμε τη διεύθυνση δικτύου που θα χρησιμοποιεί, με μια γραμμή της μορφής jail_<jailname>_ip="IP_address"

Σημαντική λεπτομέρεια: Πρέπει να εξασφαλίσουμε ότι η διεύθυνση δικτύου που βάζουμε έχει προστεθεί ως alias στην κάρτα δικτύου μας και είναι ενεργή.

Κι άλλη σημαντική λεπτομέρεια: Η διεύθυνση αυτή δεν πρέπει να χρησιμοποιείται από κάποια υπηρεσία στο host.

Προσέξτε πόσο εύκολο είναι να γίνει αυτό το τελευταίο, χωρίς να το καταλάβουμε: Πολλές από τις υπηρεσίες που ξεκινάνε κατά την εκκίνηση του συστήματος κάνουν bind σε όλες τις διαθέσιμες διευθύνσεις του μηχανήματος. Χαρακτηριστικά παραδείγματα είναι ο Apache και το sshd. Θα πρέπει να επεξεργαστείτε τα αρχεία ρυθμισεών τους, λοιπόν, ώστε να τις υποχρεώσετε να χρησιμοποιούν **μόνο** τη διεύθυνση που προορίζεται για το host. Για παράδειγμα, για τον Apache ανοίξτε το αρχείο

```
/usr/local/etc/apache22/httpd.conf
```

κι αλλάξτε τη γραμμή

```
Listen 80
```

σε

```
Listen 192.168.0.3:80
```

Για το sshd, αλλάξτε το αρχείο

```
/etc/ssh/sshd_config
```

και προσθέστε τη γραμμή

```
ListenAddress 192.168.0.3
```

όπου, προφανώς, το 192.168.0.3 είναι η διεύθυνση του host. Ίσως χρειαστεί να επανεκκινήσετε τις υπηρεσίες, ώστε να απελευθερώσουν τις άλλες διευθύνσεις:

```
# service apache22 restart
# service sshd restart
```

Καθώς καταλαβαίνετε, αν αποφασίσετε να εκτελέσετε κάποια από αυτές τις υπηρεσίες στο jail σας θα πρέπει να κάνετε αντίστοιχες ρυθμίσεις ώστε να εκτελούνται **μόνο** στη διεύθυνση δικτύου alias του jail. Ελπίζω να μην έχετε πλέον καμιά αμφιβολία γιατί οι υπηρεσίες στο FreeBSD λέγονται δαίμονες!

```
jail_pkgbuilder_devfs_enable="YES"
```

Η παραπάνω γραμμή είναι φυσικά για να γίνεται και η προσάρτηση του devfs στο jail μας. Έχοντας φροντίσει όλες αυτές τις ενοχλητικές λεπτομέρειες, είναι ώρα να ξεκινήσουμε το ωραίο μας jail. "Fingers crossed", λοιπόν, και πάμε:

```
# service jail onestart pkgbuilder
```

Χρειαζόμαστε το onestart αντί για το start, καθώς έχουμε βάλει jail_enable="NO" στο /etc/rc.conf. Αν όλα πάνε καλά θα δούμε:

```
Configuring jails:.
Starting jails: pkgbuilder.
```

Ας επιβεβαιώσουμε την εκτέλεσή του με την jls:

```
# jls
JID  IP Address     Hostname       Path
1    192.168.0.10   pkgbuilder     /scratch/jailroot
```

Μπορούμε τώρα να συνδεθούμε στο jail μας από το host, ανοίγοντας ένα shell με την εντολή jexec. Η jexec μπορεί να χρησιμοποιηθεί για να δώσουμε στο jail κάτι να τρέξει. Μη ξεχνάτε ότι το jail μας δεν έχει bash!

```
# jexec 1 tcsh
```

όπου 1 είναι το jailid (JID) που βρήκαμε με την jls.

```
root@pkgbuilder:/ #
```

Επιτυχία! Είμαστε μέσα στο jail, ως root. Τι μπορούμε να κάνουμε; Τι θα λέγατε σε πρώτη φάση να προσθέσουμε users και να ενεργοποιήσουμε τον sshd; Χρησιμοποιήστε την εντολή adduser:

```
root@pkgbuilder:/ # adduser
Username: user
Full name: test user
Uid (Leave empty for default):
Login group [user]: wheel
Login group is wheel. Invite user into other groups? []:
Login class [default]:
Shell (sh csh tcsh nologin) [sh]: tcsh
...
```

Με την ευκαιρία, αλλάξτε φυσικά και το root password γράφοντας, απλά, την εντολή passwd. Το default password είναι κενό!

Φτιάξαμε έναν χρήστη, ας δούμε τώρα τι περιέχει το /etc/rc.conf. Το ανοίγουμε με έναν editor (τον ee, αν δεν θέλετε τον vi) και, ω του θαύματος, είναι άδειο! Μην ξεχνάτε, το /etc/rc.conf στο FreeBSD περιέχει απλώς overrides για το /etc/defaults/rc.conf και το φτιάχνει το sysinstall/bsdinstall κατά την εγκατάσταση. Σε μας απλά δεν υπάρχει! Θα μπορούσατε φυσικά να αντιγράψετε διάφορες ρυθμίσεις σας από το /etc/rc.conf του host, αλλά εδώ θέλουμε απλά να ενεργοποιήσουμε τον sshd και να πειράξουμε λίγο το δίκτυο. Προσθέστε, λοιπόν, τη γραμμή

```
sshd_enable="YES"
```

Και με την ευκαιρία, προσθέστε και τη γραμμή defaultrouter ώστε να δείχνει στον router σας κι έτσι το jail να έχει πρόσβαση στο Internet:

```
defaultrouter="192.168.0.250"
```

Θα χρειαστεί επίσης να βάλετε τον κατάλληλο DNS server στο αρχείο /etc/resolv.conf που θα δημιουργήσετε. Μπορείτε, π.χ., να βάλετε τη διεύθυνση του ίδιου του router, κάποιον DNS της Google (8.8.8.8) ή έναν του ISP σας:

```
nameserver="192.168.0.250"
```

Αλλάξτε τέλος το /etc/ssh/sshd_config:

```
ListenAddress 192.168.0.10
```

Ξαναξεκινήστε όλο το δίκτυο με την εντολή /etc/netstart. Ίσως δείτε διάφορα μηνύματα λάθους, αλλά αν οι ρυθμίσεις είναι σωστές μπορείτε να τα αγνοήσετε. Οφείλονται στο γεγονός ότι το netstart προσπαθεί να κάνει κάποιες ρυθμίσεις που δεν επιτρέπονται μέσα στο jail.

Ξεκινήστε τον δαίμονα του SSH:

```
root@pkgbuilder:/ # service sshd start
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is...
```

Μπορείτε τώρα να κάνετε exit από την jexec και να συνδεθείτε στο jail σας μέσω ssh!

```
root@pkgbuilder:/ # exit
# ssh user@192.168.0.10
```

```
Password for user@pkgbuilder:
FreeBSD 9.2-RELEASE (GENERIC) #0 r255898: Thu Sep 26 22:50:31 UTC 2013
Welcome to FreeBSD...
```

Καθώς βάλαμε τον χρήστη μας στο wheel group, μπορούμε να γίνουμε root και να τρέξουμε το freebsd-update!

```
user@pkgbuilder:~ % su -
Password:
root@pkgbuilder:~ % freebsd-update fetch install
Looking up update.FreeBSD.org mirrors... 5 mirrors found.
Fetching metadata signature for 9.2-RELEASE from update5.freebsd.org... done.
Fetching metadata index... done.
```

Θέλετε να εγκαταστήσετε κάποιο port; Κανένα πρόβλημα:

```
root@pkgbuilder:~ % portsnap fetch extract
```

Βάλτε και τις παρακάτω γραμμές στο /etc/make.conf που θα φτιάξετε:

```
WITH_PKGNG="YES"
WITH_NEW_XORG="YES"
WITH_KMS="YES"
```

Μη παραξενεύεστε με το Xorg. Θα καταλάβετε παρακάτω!

```
root@pkgbuilder:~ % cd /usr/ports/shells/bash
root@pkgbuilder:~ % make install clean
root@pkgbuilder:~ % exit
user@pkgbuilder:~ % chsh -s bash
```

Ο χρήστης user του jail, απέκτησε BASH shell. Φαντάζεστε βέβαια τι πρέπει να κάνετε, προκειμένου να εγκαταστήσετε και να ρυθμίσετε τον Apache, ε; :D

Package building στη φυλακή

Δεν ονομάσαμε τυχαία το jail μας pkgbuilder! Όλοι ξέρουμε τη μανία που έχουν οι χρήστες του FreeBSD με το package building. Αν λοιπόν θέλετε να κάνετε package building αλλά δεν θέλετε ή δεν μπορείτε να χρησιμοποιήσετε το poudriere (το οποίο απαιτεί ZFS, βλ. και http://deltahacker.gr/?p=7358), μπορείτε να το κάνετε μέσα σε ένα jail. Ουσιαστικά, θα κάνετε build και install το port στο jail και μετά θα δημιουργείτε απ' αυτό τα πακέτα που μπορείτε έπειτα να εγκαταστήσετε στο host ή σε όποιο άλλο σύστημα θέλετε!

Για παράδειγμα:

```
root@pkgbuilder:~ % mkdir ~/packages
root@pkgbuilder:~ % cd ~/packages
root@pkgbuilder:~ % pkg create -a
```

Στον κατάλογο /root/packages θα βρείτε πακέτα τύπου pkgng για όλα τα ports που εγκαταστήσατε στο jail σας. Μπορείτε να τα πάρετε πολύ εύκολα: Μην ξεχνάτε, από το host έχετε πλήρη πρόσβαση στον κατάλογο του jail. Σε ένα άλλο σύστημα θα τα εγκαταστήσετε με την εντολή pkg add. Καταλάβατε, φαντάζομαι, γιατί βάλαμε ρυθμίσεις για το Xorg σε ένα jail που δεν θα τρέξει ποτέ περιβάλλον γραφικών!

Ολοκληρώνοντας...

Μα, δεν τελειώνουν τα jails, έχετε τόσο διάβασμα να κάνετε! Ξεκινήστε από το FreeBSD Handbook φυσικά

http://freebsd.org/doc/handbook/jails-build.html

όπου θα βρείτε πολλά ευχάριστα απογεύματα, εχμ, εννοούμε πληροφορίες για το πώς να φτιάξετε μεταξύ άλλων service jails και να διαχειριστείτε τα jails σας μέσω διάφορων sysctl tunables. Ελπίζουμε να βρείτε χρήσιμες κι ωραίες εφαρμογές για το νέο σας παιχνίδι!



Skill: Beginner
Tags: Bitcoin, Bitcoin difficulty, Bitcoin mining pool, FPGA, ASIC



Φρέσκια παραγωγή Bitcoins, από σύγχρονους χρυσοθήρες!

Τι είναι το Bitcoin mining; Μπορώ να το κάνω με τον υπολογιστή μου; Θέλω να βγάλω εύκολα και γρήγορα λεφτά, μπορώ; Ποια αξίνα να αγοράσω και σε ποιο βουνό ν' ανέβω; Αυτές είναι μερικές από τις απορίες που έχουν όσοι έρχονται για πρώτη φορά σ' επαφή με το Bitcoin γενικότερα και το Bitcoin mining ειδικότερα. Ας δούμε και τις απαντήσεις...

Στο άρθρο "Bitcoin: Αληθινό νόμισμα ή ροζ τσιχλόφουσκα με πράσινες βούλες;", το οποίο φιλοξενήσαμε στο deltaHacker 008, κάναμε μια *μικρή* εισαγωγή στο cryptocurrency του Bitcoin και είδαμε πώς μπορούμε να το χρησιμοποιήσουμε κι εμείς. Τώρα θα μπούμε στα ενδότερα του Bitcoin συνεισφέροντας στο αντίστοιχο δίκτυο, με το αζημίωτο φυσικά. Ας ξεκινήσουμε από τα βασικά: Τι είναι το Bitcoin mining; Μη νομίζετε ότι αφορά στο σκάψιμο της Γης με κάποιο εργαλείο. Με αυτόν τον όρο αναφερόμαστε στην εργασία που πρέπει να ολοκληρώσει κάποιος, προκειμένου να επιβεβαιώσει μερικές συναλλαγές στο δίκτυο του Bitcoin και να τις προσθέσει στην τεράστια λίστα των ολοκληρωμένων/έγκυρων συναλλαγών. Για να προστεθεί ένα νέο block συναλλαγών στο chain block του δικτύου του Bitcoin, πρέπει να λυθεί ένα αρκετά πολύπλοκο μαθηματικό πρόβλημα. Η επίλυση αυτού του προβλήματος είναι τόσο επίπονη, που κατέληξε να παρομοιάζεται με το αληθινό σκάψιμο. Οι Bitcoin miners συναγωνίζονται μεταξύ τους για το ποιος θα βρει πρώτος τη λύση, ώστε να προσθέσει το αντίστοιχο block στην αλυσίδα των ήδη υπολογισμένων blocks. Τέλος, σημειώστε ότι με την προσθήκη ενός νέου block παίρνουμε μια αμοιβή σε Bitcoin[1].



Το Bitcoin κάποιοι πιστεύουν ότι αποτελεί το μέλλον του χρήματος, τη στιγμή που κάποιοι άλλοι το αντιμετωπίζουν ως μια τεράστια φούσκα...

Θα πλουτίσω;

Προηγουμένως αναφερθήκαμε στα blocks και στην αμοιβή που λαμβάνει κανείς όταν υπολογίζει ένα block. Ίσως με όλα αυτά να σας δημιουργήσαμε την εντύπωση ότι όσο περισσότερες συναλλαγές επιβεβαιώνουμε, τόσα περισσότερα blocks προσθέτουμε στο chain block του δικτύου, επομένως τόσο περισσότερα χρήματα παίρνουμε. Σωστά; Λάθος!

Σκεφτείτε τι θα γινόταν αν μπορούσαμε να υπολογίσουμε νέα Bitcoin blocks με ευκολία. Θα γινόμασταν όλοι πλούσιοι και το δίκτυο θα κατέρρεε. Για να μη συμβεί κάτι τέτοιο, υπάρχει ένας μηχανισμός που ρυθμίζει τη δυσκολία υπολογισμού των blocks. Αυτός ο μηχανισμός λειτουργεί αυτόματα, ρυθμίζεται συλλογικά από το δίκτυο[2] κι επηρεάζει τους πάντες. Όσο αυξάνονται οι χρήστες και η διαθέσιμη υπολογιστή ισχύς στο δίκτυο, τόσο αυξάνει και το λεγόμενο Bitcoin difficulty. Όπως καταλαβαίνετε, με τη βελτίωση της τεχνολογίας και την κυκλοφορία νεότερων και ισχυρότερων μηχανημάτων, αυτό το μέγεθος αυξάνεται. Έτσι, με την πάροδο του χρόνου είναι όλο

1. Αρχικά ήταν 50 Bitcoins αλλά πλέον είναι 25. Η αμοιβή αυτή μειώνεται περίπου κάθε 210,000 blocks (περίπου 4 ημερολογιακά έτη).
2. Όχι απ’ τους χρήστες, αλλά από τον αλγόριθμο του Bitcoin!

και πιο δύσκολο να πλουτίσει κανείς, εργαζόμενος ως Bitcoin miner. Όλα αυτά όμως δεν σημαίνουν ότι το Bitcoin difficulty ακολουθεί μονίμως ανοδική πορεία. Όλο και *κάτι*[3] μπορεί να συμβεί, με συνέπεια η "δυσκολία" του δικτύου να πέσει ακόμα και στο μισό :D Πάντως, πέρα από τα διάφορα σενάρια, πρέπει να πούμε το εξής: Ακόμα κι αν πάρουμε ένα αρκετά δυνατό μηχάνημα, με τα σημερινά δεδομένα θα βγάζουμε περίπου 0,01 BTC (Bitcoin) την μέρα.

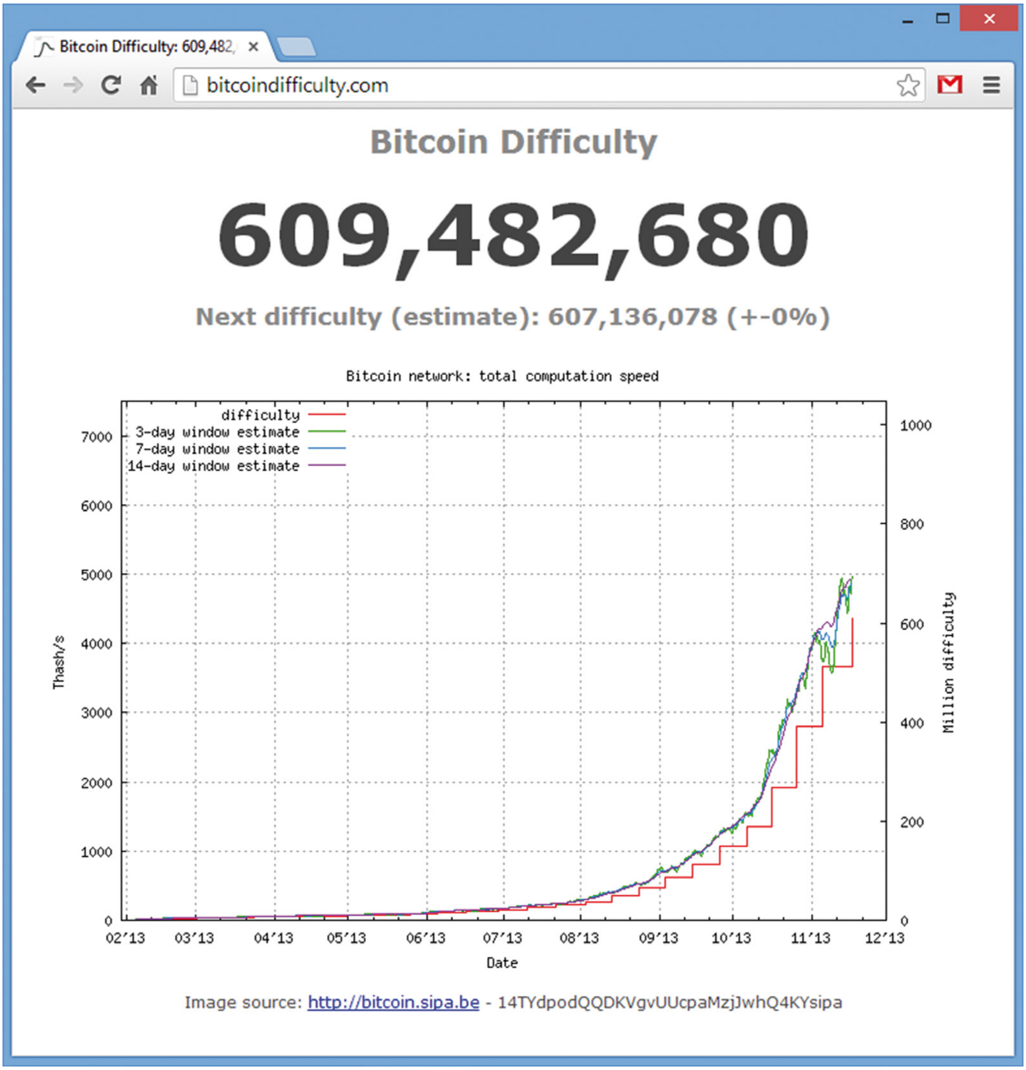
Σκάψιμο με στιλ

Προφανώς και *δεν* θα σκάψουμε εμείς οι ίδιοι. Όλες οι εργασίες, από τις πιο απλές μέχρι τις πιο σύνθετες, θα γίνουν αυτόματα από ένα πρόγραμμα στον υπολογιστή μας. Όμως, πριν λερώσουμε τα χέρια μας με τα ψηφιακά χώματα (Σ.τ.Α. Δεν γλυτώνουμε εύκολα απ' αυτή τη μεταφορά :P) πρέπει να ξεκαθαρίσουμε τι προσπαθούμε να πετύχουμε και ποια τακτική ακολουθούμε. Όπως αντιλαμβάνεστε, θα συνδεθούμε ξανά στο δίκτυο του Bitcoin, αλλά αυτή τη φορά ως miners και όχι ως απλοί χρήστες. Συγκεκριμένα, θα συνεισφέρουμε στη λειτουργία του δικτύου χαρίζοντας επεξεργαστική ισχύ. Στο παρόν άρθρο δεν θα ασχοληθούμε με τη φύση των σχετικών υπολογισμών, αλλά με το πώς μπορούμε να επιταχύνουμε την ολοκλήρωσή τους. Εν ολίγοις, θα δούμε πώς μπορούμε να γίνουμε αποτελεσματικοί στο σκάψιμο, ώστε να αυξήσουμε τα κέρδη μας.



Μέχρι πριν από μερικά χρόνια τα Bitcoin mining rigs απαρτίζονταν από δεκάδες κάρτες γραφικών, που έκαναν αριθμητικές πράξεις ακατάπαυστα, παρήγαγαν άφθονη θερμότητα κι έκαιγαν ρεύμα στο φουλ!

3. Για παράδειγμα, ένα DDoS attack στο μεγαλύτερο Bitcoin mining pool, όπου εργάζονται πολλοί miners συλλογικά.

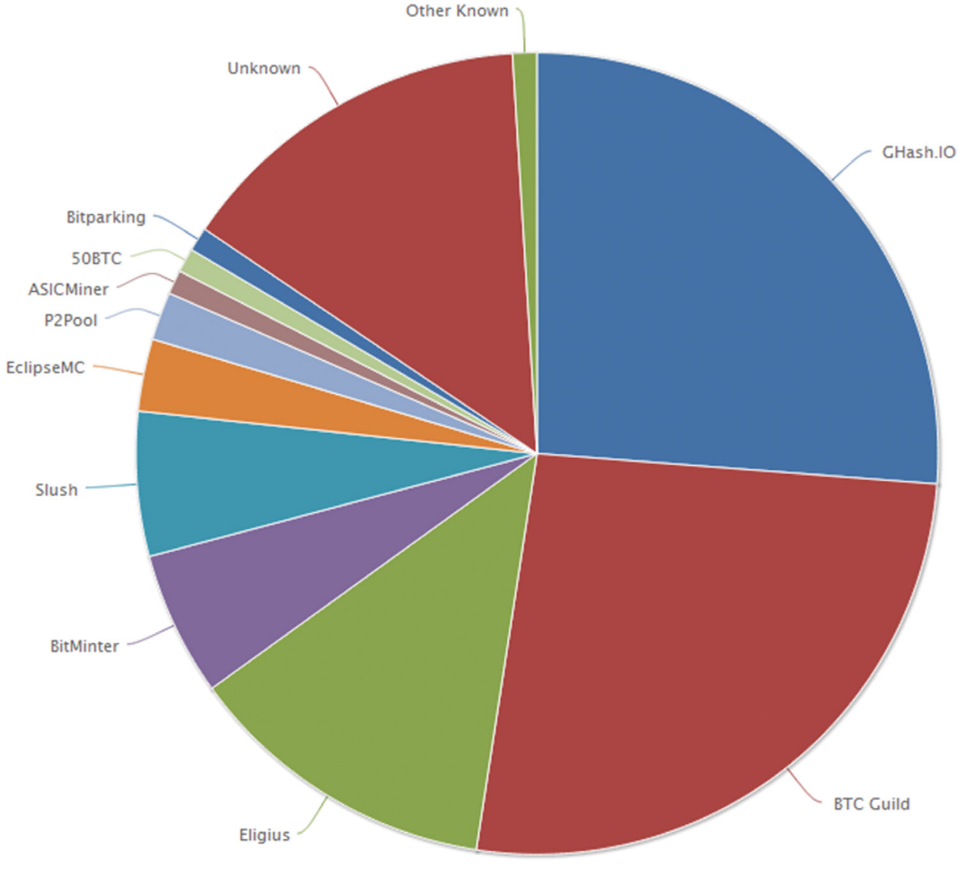


Το Bitcoin difficulty έχει φτάσει σε ύψος ρεκόρ τους τελευταίους μήνες. Αυτό οφείλεται στις νέες συσκευές ASIC, που χάρη στην τεράστια επεξεργαστική τους ισχύ διευκολύνουν τον υπολογισμό των blocks και... αυξάνουν τον πληθωρισμό!

Ποια αξίνα να πάρω;

Εδώ χρειάζεται αρκετή μελέτη (την οποία έχουμε κάνει ήδη :D). Σε γενικές γραμμές, χρειαζόμαστε κάτι φθηνό, το οποίο θα μπορούμε να λειτουργούμε για πολλές μέρες ασταμάτητα κι από το οποίο θα καταφέρουμε τελικά να βγάλουμε κάτι. Μέχρι πριν από αρκετούς μήνες, η τάση στον κόσμο του Bitcoin ήταν να χρησιμοποιούνται υπολογιστές σαν αυτούς που χρησιμοποιούμε καθημερινά. Ένας τέτοιος υπολογιστής, με τη βοήθεια μιας κάρτας γραφικών AMD 6990, ήταν σε θέση να υπολογίζει γύρω στα 700MH/s[4]. Γεμίζοντας τις θύρες επέκτασης με τέσσερις τέτοιες κάρτες, αποκτού-

4. Mega Hashes / second: Πρόκειται για μια μονάδα μέτρησης της υπολογιστικής ισχύς, που χρησιμοποιείται στον κόσμο του Bitcoin mining.



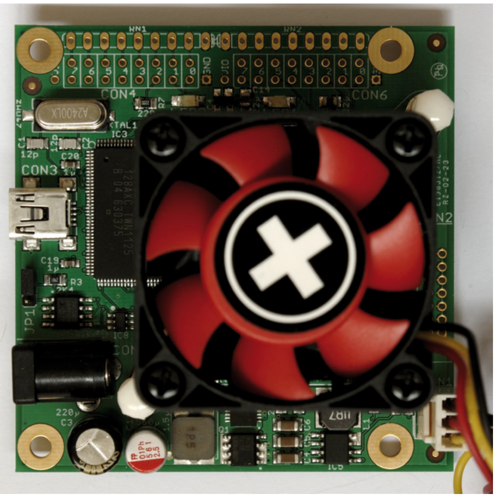
Τα Bitcoin mining pools αποτελούν υπηρεσίες στις οποίες οι χρήστες εργάζονται ομαδικά για τον υπολογισμό των blocks. Τα κέρδη μοιράζονται ανάλογα με το ποσοστό συμμετοχής του καθενός.

σε κανείς ένα πολύ ισχυρό μηχάνημα, από το οποίο έβγαζε αρκετά Bitcoins. Πλέον, τα μηχανήματα αυτά είναι εντελώς ασύμφορα. Βλέπετε, με τα σημερινά δεδομένα, το κόστος λειτουργίας ενός τέτοιου μηχανήματος είναι μεγαλύτερο από το κέρδος του ιδιοκτήτη σε Bitcoin. Κάπως έτσι έκαναν την εμφάνισή τους οι FPGA miners. Τα FPGA είναι προγραμματιζόμενα τσιπάκια, αλλά δεν έχουν καμία σχέση με τους μικροελεγκτές. Όταν προγραμματίζουμε ένα FPGA, αλλάζουμε την εσωτερική δομή του ίδιου του hardware! Δεν παραθέτουμε εντολές προς εκτέλεση, αλλά περιγράφουμε τη νέα δομή που θέλουμε να αποκτήσει το κύκλωμα του τσιπ. Ακριβώς γι' αυτό τα FPGA είναι πολύ δύσχρηστα, αλλά προσφέρουν τεράστια ισχύ και αξιοποιούνται σε πολλές συσκευές και με πολλούς τρόπους. Για παράδειγμα, υπάρχουν FPGA που λειτουργούν σαν μηχανές Arcade όπως και FPGA που αποκωδικοποιούν σήματα HDMI. Στη δική μας περίπτωση, τα FPGA αξιοποιήθηκαν για να κάνουν τους υπο-

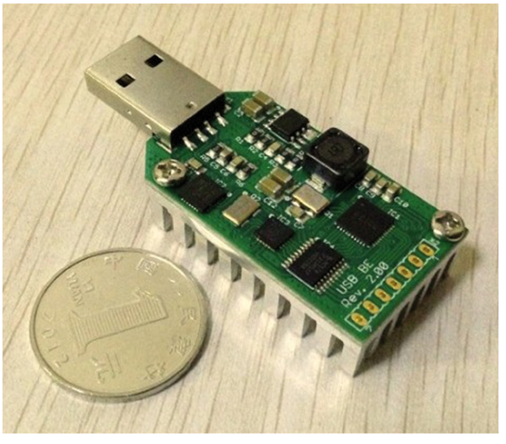
λογισμούς που περιλαμβάνει το Bitcoin mining. Για να κατανοήσετε ποιο ήταν το όφελος από τη χρήση τους, σκεφτείτε ότι ένα FPGA καταφέρνει να υπολογίζει 700 με 800MH/s (όσα και μια ισχυρή κάρτα γραφικών), αλλά η τιμή του είναι πολύ χαμηλή και η κατανάλωση ελάχιστη! Μην νομίζετε όμως ότι οι miners έμειναν για πολύ καιρό στα FPGA. Τα εν λόγω τσιπάκια άνοιξαν το δρόμο, ώστε όλοι οι σχετικοί υπολογισμοί να πραγματοποιούνται από hardware κι όχι από software. Ωστόσο, λίγο καιρό μετά την εμφάνιση των FPGA miners, άρχισαν να εμφανίζονται οι λεγόμενοι ASIC miners! Τα ASIC είναι ολοκληρωμένα κυκλώματα που έχουν σχεδιαστεί και κατασκευαστεί κατά παραγγελία και σύμφωνα με τις συγκεκριμένες απαιτήσεις της εκάστοτε εφαρμογής. Η διαφορά μεταξύ των ASIC και των FPGA θυμίζει τη διαφορά μεταξύ των interpreted και compiled προγραμμάτων. Τα interpreted προγράμματα είναι εύκολα τροποποιήσιμα αλλά "αργόστροφα", ενώ τα compiled προγράμματα δεν τροποποιούνται καθόλου αλλά από την άλλη εκτελούνται ταχύτατα! Με τη βοήθεια των ASIC μπορούμε να δημιουργήσουμε συσκευές με ισχύ που ούτε καν φανταζόμασταν παλιότερα. Για να καταλάβετε τι εννοούμε, σκεφτείτε ότι οι πρώτες συσκευές ASIC υπολόγιζαν περί τα 2.5GH/s, με κατανάλωση μικρότερη από 10W.

Έρευνα αγοράς

Σκοπός μας είναι να ασχοληθούμε με το Bitcoin και *αν* τα καταφέρουμε, να βγάλουμε έστω τα έξοδα για την αγορά και τη λειτουργία της απαιτούμενης συσκευής. Για αυτό το λόγο θα στραφούμε αποκλειστικά στα μηχανήματα ASIC! Δεν έχει κανένα νόημα να επενδύσουμε σε τεχνολογίες που θεωρούνται ξεπερασμένες για αυτή τη δουλειά. Ένας αρκετά καλός FPGA miner αποδίδει πολύ χειρότερα από τον φθηνότερο ASIC miner, ενώ οι τιμές τους κυμαίνονται περίπου στα ίδια επίπεδα. Το ποια εταιρεία θα επιλέξουμε



Ένας FPGA miner καταφέρνει να υπολογίσει περίπου τα ίδια hashes ανά δευτερόλεπτο που υπολογίζει και μια high-end κάρτα γραφικών. Η διαφορά τους εντοπίζεται στο κόστος αγοράς και φυσικά στο κόστος λειτουργίας.



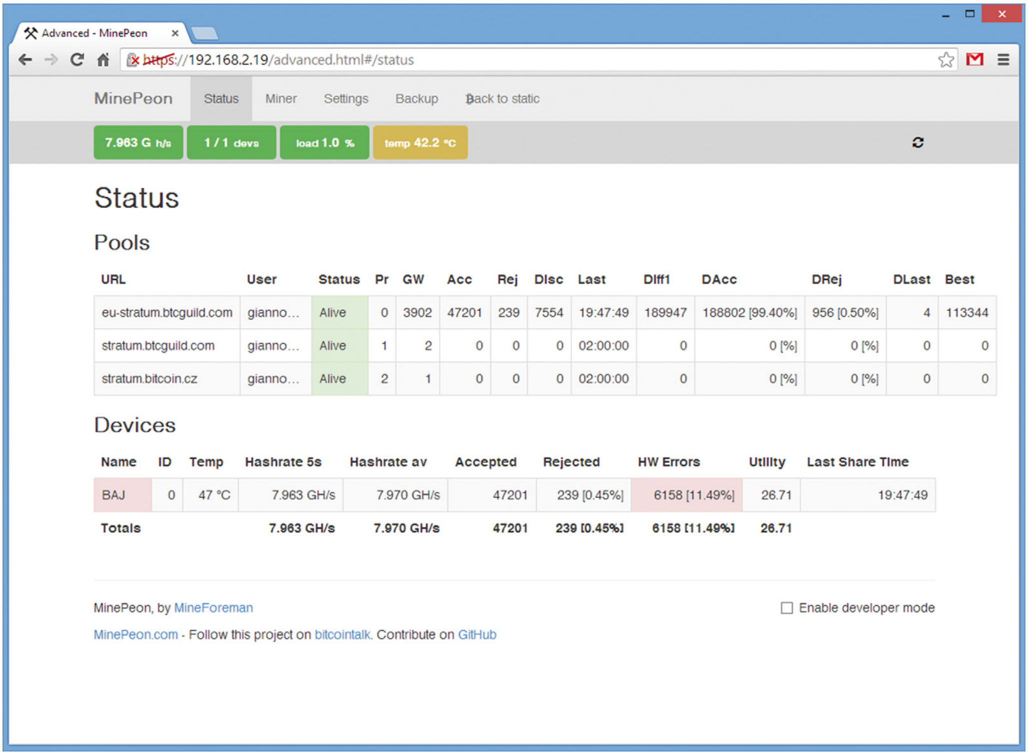
Οι ASIC miners καταφέρνουν να υπολογίσουν υπερπολλαπλάσια hashes ανά δευτερόλεπτο, με ακόμα μικρότερη κατανάλωση. Ο miner της φωτογραφίας καταφέρνει να υπολογίσει περίπου 400MH/s, ενώ τροφοδοτείται μόνο από τη θύρα USB.



Με τα πολλά καταλήξαμε στο Butterfly Labs Bitforce SC, το οποίο υπολογίζει γύρω στα 5GH/s. Κατά πάσα πιθανότητα πρόκειται για την πιο φθηνή λύση σε ASIC, με την οποία αξίζει να ασχοληθεί κάποιος αρχάριος.

βασίζεται περισσότερο σε εκτιμήσεις και λιγότερο σε χειροπιαστά δεδομένα. Ποια εταιρεία πιστεύουμε ότι πετυχαίνει την καλύτερη σχέση τιμής με GH/s; Ποια εταιρεία θα διαθέσει τις συσκευές της την ημερομηνία που ανακοινώθηκε εξ αρχής; Ποια εταιρεία έχει το λιγότερο προβληματικό hardware; Όλες αυτές είναι ερωτήσεις που δύσκολα απαντώνται, αλλά μπορούμε να πάρουμε μια γεύση διαβάζοντας κριτικές στα διάφορα fora στο Internet. Εμείς εμπιστευτήκαμε την εταιρεία ButterflyLabs και συγκεκριμένα το BitForce SC ASIC miner. Δυστυχώς, όμως, αν και αγοράστηκε στα τέλη του Μάρτη του 2013, έφτασε στα χέρια μας μερικές εβδομάδες πριν[5]. Υπάρχουν κι άλλες εταιρείες που αξίζει να τσεκάρουμε. Ενδιαφέρον παρουσιάζουν και οι συσκευές Block Erupter. Υπολογίζουν μόνο 330MH/s, αλλά κοστίζουν μόλις 0,07 BTC.

5. Με μερικούς πρόχειρους υπολογισμούς, αν ξεκινούσαμε τότε το mining, θα βγάζαμε γύρω στα 700 δολάρια μέσα στον πρώτο μήνα. Λόγω της καθυστέρησης και της αύξησης τους Bitcoin difficulty, είναι πιθανό να μην βγάλουμε ποτέ τα έξοδα :D



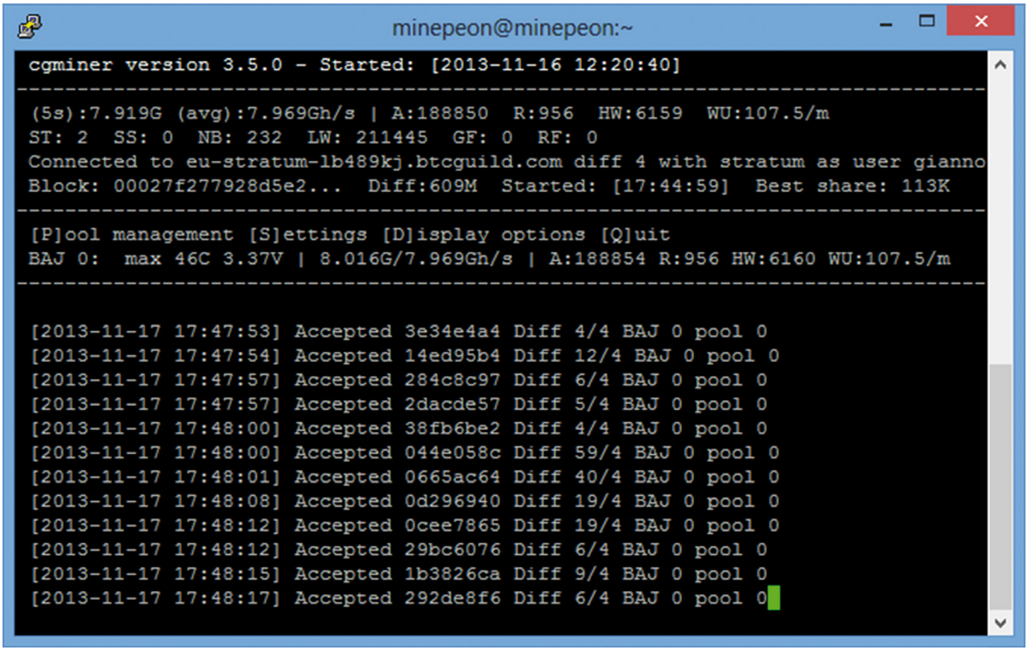
Το Minepeon είναι μια έκδοση του Arch Linux προσαρμοσμένη στο Raspberry Pi, που διαθέτει όλα τα εργαλεία για το Bitcoin mining καθώς και ένα εύχρηστο web UI. Το μόνο που χρειάζεται να κάνουμε εμείς είναι να δώσουμε τα στοιχεία μας και να επιλέξουμε ένα mining pool.

Αν δεν φιλοδοξούμε να πλουτίσουμε από το Bitcoin mining και δεν έχουμε πρόβλημα να ξοδέψουμε 10 δολάρια, οι εν λόγω συσκευές αποτελούν μια πολύ καλή ευκαιρία.

Το δικό μας mining rig

Στα δεξιά μας κάθεται το νέο, σύγχρονο —κι επιτρέψτε μας την έκφραση— "τούμπανο" ASIC. Τα περισσότερα μηχανήματα του είδους, εκτός από κάποια που στοιχίζουν δεκάδες χιλιάδες ευρώ, απαιτούν τη σύνδεση με κάποιον υπολογιστή. Εμείς, στη θέση αυτού του υπολογιστή χρησιμοποιήσαμε το Raspberry Pi. Βλέπετε, δεν έχει κανένα νόημα να αφήνουμε τον κύριο υπολογιστή μας αναμμένο, ολημερίς κι ολονυχτίς. Αν κάναμε κάτι τέτοιο, ο υπολογιστής θα κατανάλωνε ένα καλό μερίδιο απ' τα κέρδη μας!

Το μόνο που χρειαζόμαστε είναι το εργαλείο cgminer, το οποίο αναλαμβάνει τα πάντα. Πρόκειται για ένα πρόγραμμα Bitcoin mining, που υποστηρίζει μια τεράστια γκάμα συσκευών, από τον πιο φθηνό FPGA μέχρι τον πιο ισχυρό ASIC miner. Θα μπορούσαμε να το εγκαταστήσουμε πάνω στο Raspbian, αλλά, μας ξέρετε, ψάχνουμε πάντα για το κάτι παραπάνω ;) Μετά λοιπόν από καμπόσο ψάξιμο καταλήξαμε στο Minepeon. Πρόκειται για μια διανομή που προορίζεται εξαρχής για σκάψιμο. Κατεβάσαμε τη νεότερη έκδοση, την περάσαμε στην κάρτα SD του Raspberry Pi κι όταν



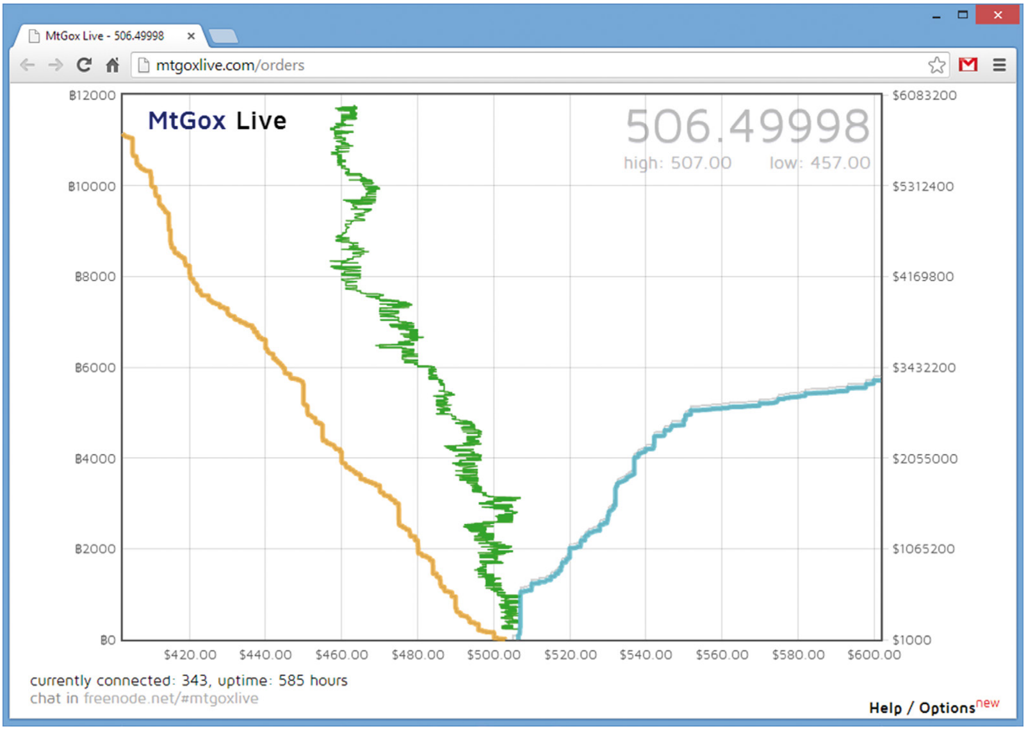
Αν δεν μας αρέσουν τα φανταχτερά UI, μπορούμε να χρησιμοποιήσουμε τον miner και από την γραμμή εντολών μέσω SSH.

ενεργοποιήθηκε το σύστημα συνδέσαμε το ASIC. Χρησιμοποιώντας τη διεύθυνση IP του Raspberry Pi, μπορούμε να συνδεθούμε σε ένα απλό αλλά αρκετά εύχρηστο web interface, μέσω του οποίου έχουμε την ευκαιρία να ρυθμίσουμε διάφορες παραμέτρους του miner. Η σημαντικότερη από αυτές σχετίζεται με την επιλογή κάποιου Bitcoin mining pool. Αυτές οι υποδομές (τα mining pools) επιτρέπουν τη συλλογική εργασία μεταξύ διαφορετικών χρηστών, με στόχο την αύξηση της παραγωγικότητας και των κερδών. Οι χρήστες ενός pool εργάζονται ομαδικά και μοιράζονται τα όποια κέρδη, ανάλογα με την ισχύ που προσέφερε ο καθένας. Εδώ πρέπει να προσέξουμε το ποσοστό επί των κερδών που κρατάει το pool, για να καλύψει τα έξοδα λειτουργίας του. Πάντως, όπως και με τις εταιρείες κατασκευής ASIC, δεν υπάρχουν pools που είναι (αποδεδειγμένα) καλά ή κακά. Η συμμετοχή σε κάποιο αποτελεί προσωπική επιλογή του καθενός και βασίζεται κυρίως σε υποκειμενικές εκτιμήσεις. Μια καλή αρχή αποτελεί το BTC Guild[6].

Εδώ το καλό Bitcoin!

Ας υποθέσουμε ότι καταφέραμε να έχουμε ένα μικρό κι ενδεχομένως σταθερό[7] εισόδημα από Bitcoins. Τι μπορούμε να κάνουμε με αυτά; Μπορούμε να τα αποταμιεύσουμε σε κάποιο εικονικό πορτοφόλι (Bitcoin wallet) ή να τα αξιοποιήσουμε άμεσα, για την πραγματοποίηση αγορών. Σε γενικές γραμμές, με τα σκαμπανεβάσματα που παρατηρούνται στην ισοτιμία μεταξύ Bitcoin και Δολαρίου, αξίζει να τα κρατήσουμε και να μη βιαστούμε να τα ξοδέψουμε. Τα Bitcoins, όμως, μπορούμε και να τα πουλήσουμε! Υπάρχουν αρκετά Bitcoin marketplaces, όπου μπορεί ο κα-

6. …Αν και αυτό αποτελεί προϊόν προσωπικών εκτιμήσεων ;-)
7. Για τις επόμενες 10 μέρες, το πολύ!



Παρακολουθώντας ζωντανά την ισοτιμία μεταξύ BTC και USD, περιμένουμε την κατάλληλη στιγμή για να πουλήσουμε τα Bitcoins (που δεν έχουμε :D)

θένας να πουλήσει και να αγοράσει Bitcoins. Εφόσον ενδιαφερόμαστε να πουλήσουμε, πρέπει να εξασφαλίσουμε ότι τη στιγμή της αγοραπωλησίας η τιμή θα είναι όσο το δυνατόν υψηλότερη. Ένα από τα μεγαλύτερα marketplaces είναι το Mt.Gox (`https://mtgox.com`). Εμείς πάντως δεν το έχουμε χρησιμοποιήσει ακόμα, γιατί περιμένουμε τη συγκέντρωση του αστρονομικού ποσού του ενός Bitcoin! Όταν τα καταφέρουμε, όμως, το Mt.Gox θα αποτελέσει την πρώτη μας επιλογή.

Είναι τόσο εύκολο να πλουτίσω;

Θα μπορούσαμε να συγκρίνουμε το Bitcoin mining με κάποια επένδυση εντός ή και εκτός του Internet. Για αρχή, το μόνο σίγουρο είναι ότι θα πρέπει να *δώσουμε* χρήματα. Το πόσα θα δώσουμε καθορίζεται από την τόλμη και φυσικά από την τσέπη μας. Στη συνέχεια, το αν θα βγάλουμε κέρδος ή όχι, καθορίζεται από πολλούς παράγοντες και δεν είναι σίγουρη ούτε καν η απόσβεση του αρχικού κεφαλαίου. Μήπως αυτό το πράγμα δεν είναι επένδυση, αλλά τζόγος; Η διαχωριστική γραμμή είναι ούτως ή άλλως αρκετά λεπτή! Η αξία του Bitcoin σε ευρώ είναι δύσκολο να προβλεφθεί και το ίδιο ισχύει για το Bitcoin difficulty, το οποίο κανένας δεν γνωρίζει πώς θα κυμανθεί μέσα στις επόμενες εβδομάδες. Μήπως τελικά θα χρειαστεί να παρακολουθούμε *και* τις εταιρείες που βγάζουν αδιάκοπα νέο και καλύτερο hardware; Εμείς πάντως το βλέπουμε μόνο σαν ένα οριακά επικερδές χόμπι και ήμαστε ήσυχοι.

Skill: Intermediate
Tags: LAMP, PHP, web, programming, object, class, include, require, input, sanitization, filter



Web Development σε περιβάλλον LAMP [μέρος 3]

Σ' αυτό το μέρος της σειράς θα γνωρίσουμε την αντικειμενοστραφή πλευρά της PHP. Επιπρόσθετα θα μάθουμε να ενσωματώνουμε αρχεία κώδικα μέσα σε άλλα και θα εξετάσουμε τις μεθόδους εισαγωγής δεδομένων από το χρήστη, μέσα από φόρμες HTML. Φυσικά θα δούμε πώς γίνεται και το σχετικό φιλτράρισμα, ώστε να απορρίπτονται οι ανεπιθύμητοι κι ενδεχομένως δόλια τοποθετημένοι χαρακτήρες.

Η PHP, όπως οι περισσότερες σύγχρονες γλώσσες προγραμματισμού, είναι αντικειμενοστραφής (object oriented). Υποστηρίζει τον ορισμό κλάσεων, μεθόδων και ιδιοτήτων, τη δημιουργία αντικειμένων και, γενικότερα, οτιδήποτε περιμένει κανείς από μια αντικειμενοστραφή γλώσσα. Στη σειρά άρθρων για την εκμάθηση της Python (βλ. deltaHacker 005 έως και 011) ο καλός συνάδελφος Sonic2000gr παρουσίασε όλες αυτές τις έννοιες αναλυτικά, ενώ έδωσε και αρκετά παραδείγματα. Έτσι, αυτή τη φορά δεν χρειάζεται να αναλωθούμε στην επεξήγησή τους. Σημειώστε εδώ ότι υπάρχει κι εκείνη η προσφορά που περιγράφεται στο http://deltahacker.gr/pdf12order: Αν θέλετε να εμβαθύνετε σε αυτές τις έννοιες, μπορείτε να προμηθευτείτε παλιότερα τεύχη και να διαβάσατε τα καταπληκτικά άρθρα του Sonic. Μ' αυτόν τον τρόπο θα μάθετε **και** Python ;)

Θα ξεκινήσουμε την περιπλάνησή μας με το βασικότερο συστατικό: τις **κλάσεις**. Η δήλωση μιας κλάσης ξεκινά με τη λέξη κλειδί class, ακολουθούμενη από το επιθυμητό όνομα. Αμέσως μετά ανοίγει ένα μπλοκ κώδικα, εντός του οποίου ορίζονται τα συστατικά της κλάσης: Οι μέθοδοι και οι ιδιότητες. Ας δούμε ένα απλό παράδειγμα:

```
class car{
    private $color = "red" ;
    protected $doors = "5" ;

public function change_color($color){
            $this->color = $color ;
}
}
```

Μέσα στην κλάση ορίζονται οι μεταβλητές/ιδιότητες color και doors, όπως επίσης η συνάρτηση/μέθοδος change_color. Όπως θα προσέξατε, ο ορισμός όλων αυτών των στοιχείων ξεκινά με κάποια από τις λέξεις private, protected και public. Αυτές οι λέξεις ονομάζονται **access modifiers** (τροποποιητές πρόσβασης) και καθορίζουν την "περιοχή" του προγράμματος εντός της οποίας ισχύει ο εκάστοτε ορισμός. Τα στοιχεία που ορίζονται σαν **public** είναι προσβάσιμα από οπουδήποτε. Έτσι, η συνάρτηση change_color είναι προσβάσιμη κι εκτός της κλάσης. Τα στοιχεία με τον modifier private είναι προσβάσιμα αποκλειστικά εντός της κλάσης. Μόνο οι μέθοδοι της εκάστοτε κλάσης μπορούν να επιδράσουν στις τιμές των μεγεθών **private**. Επιπρόσθετα, τα στοιχεία αυτού του είδους δεν κληρονομούνται στις τυχόν υποκλάσεις. Τα στοιχεία που ορίζονται σαν **protected** είναι προσβάσιμα μόνον εντός κλάσης, αλλά σε αντίθεση με εκείνα που ορίζονται σαν private κληρονομούνται από τις τυχόν υποκλάσεις. Εάν δεν χρησιμοποιηθεί τροποποιητής για ένα στοιχείο, η PHP επιλέγει αυτόματα το public. Τέλος, κάτι ακόμα που ενδέχεται να παρατηρήσατε είναι αυτό το περίεργο "$this->color". Η λέξη κλειδί **$this** αναφέρεται πάντα στο ίδιο το αντικείμενο, στο οποίο ανήκει η εκάστοτε μέθοδος.

Για να ορίσουμε ένα αντικείμενο χρησιμοποιούμε τη λέξη κλειδί **new**:

```
$aCar = new car() ;
```

Με την παραπάνω γραμμή δημιουργείται το αντικείμενο aCar, το οποίο ανήκει στην κλάση car. Όταν καλούμε τη μοναδική μέθοδο της κλάσης car (change_color) μέσα από το αντικείμενο aCar, η λέξη κλειδί $this αναφέρεται στο aCar. Παρεμπιπτόντως, για την πρόσβαση στις ιδιότητες και στις μεθόδους ενός αντικειμένου χρησιμοποιείται ο τελεστής που θυμίζει βέλος (->):

```
$aCar->change_color("orange") ;
```

Με την παραπάνω γραμμή καλούμε τη μέθοδο change_color, του αντικειμένου $aCar.

Πολλές φορές θέλουμε να αρχικοποιήσουμε τις ιδιότητες μιας κλάσης. Για να πετύχουμε κάτι τέτοιο στην PHP, αρκεί να χρησιμοποιήσουμε τη λεγόμενη **constructor function** (συνάρτηση δημιουργίας). Πρόκειται για μια συνάρτηση που έχει πάντα το όνομα __construct()[1] και δηλώνεται μέσα στον ορισμό της αντίστοιχης κλάσης. Μια συνάρτηση δημιουργίας για την κλάση car θα μπορούσε να έχει ως εξής:

```
function __construct($color, $doors){
     $this->color = $color ;
     $this->doors = $doors ;
}
```

Παρατηρείστε ότι η συνάρτηση __construct() δέχεται κάποιες παραμέτρους. Περιλαμβάνοντας αυτή τη συνάρτηση στη δήλωση της κλάσης car, μπορούμε να ορίζουμε αντικείμενα car και ταυτόχρονα να αρχικοποιούμε τις μεταβλητές/ιδιότητές τους. Σε αυτόν τον ορισμό-αρχικοποίηση ορίζονται και οι παράμετροι της συνάρτησης __construct(). Δείτε πώς γίνεται κάτι τέτοιο και θα καταλάβετε αμέσως:

```
$aCar = new car("blue", 3) ;
```

Το αντίθετο της συνάρτησης δημιουργίας αποτελεί η λεγόμενη **destructor function** (συνάρτηση καταστροφής), η οποία έχει πάντα το ειδικό όνομα __destruct(). Η εν λόγω συνάρτηση δεν δέχεται παραμέτρους και επιτρέπει μόνο την εκτέλεση ενός τμήματος κώδικα πριν από την "καταστροφή" ενός αντικειμένου.

Ελεγχόμενη πρόσβαση

Σε γενικές γραμμές, είναι φρόνιμο να μην παρέχουμε άμεση πρόσβαση στις μεταβλητές μιας κλάσης, αλλά να χρησιμοποιούμε τις κατάλληλες μεθόδους. Είμαστε βέβαιοι ότι θα έχετε δει τέτοιες μεθόδους σε πολλές αντικειμενοστραφείς γλώσσες προγραμματισμού. Οι μέθοδοι που προορίζονται για την ανάγνωση ή την εγγραφή σε "κρυμμένες" μεταβλητές μιας κλάσης έχουν ονόματα που ξεκινούν συνήθως με τις λέξεις set ή get. Στην περίπτωση της κλάσης car, για παράδειγμα, θα μπορούσαμε να ορίσουμε τις ακόλουθες μεθόδους: getColor(), setColor(), getDoors() κ.λπ. Για την ευκολία μας, οι δημιουργοί της PHP έχουν εισαγάγει τις ειδικές συναρτήσεις

1. Πριν από το όνομα προηγούνται δύο underscore (κάτω παύλες).

__set() και __get(). Αυτές οι συναρτήσεις καλούνται αυτόματα όταν προσπαθούμε να τροποποιήσουμε ή να διαβάσουμε την τιμή μιας μεταβλητής, η οποία ανήκει στην κλάση και έχει οριστεί σαν protected ή private. Η συνάρτηση __get() δέχεται μία μόνο παράμετρο (το όνομα της μεταβλητής) και η __set() δύο (το όνομα της μεταβλητής και την επιθυμητή νέα τιμή). Μη νομίζετε όμως ότι αυτές οι συναρτήσεις παρέχονται έτοιμες από το περιβάλλον της PHP! Οι εν λόγω συναρτήσεις ορίζονται από το χρήστη, ώστε να επιλέξει ο ίδιος το όποιο "φιλτράρισμα" θέλει να γίνεται κατά την πρόσβαση στις μεταβλητές. Δείτε πώς θα μπορούσαμε να ορίσουμε τις εν λόγω συναρτήσεις για την κλάση car:

```
function __get($name){
    return $this->$name ;
}
function __set($name, $value){
    if($name == "doors"){
        if($value < 2)
            $this->doors = 2 ;
        else
            $this->doors = $value ;
    }
    $this->$name = $value ;
}
```

Η συνάρτηση __get() επιστρέφει την τιμή της μεταβλητής που προσπαθούμε να διαβάσουμε, χωρίς κανένα φιλτράρισμα ή άλλου είδους επέμβαση στην τιμή της. Αντίθετα, η συνάρτηση __set() ελέγχει αν προσπαθούμε να επέμβουμε στη μεταβλητή doors και παρεμβαίνει όταν προσπαθούμε να καταχωρήσουμε μια τιμή μικρότερη του 2. Βλέποντας τους ορισμούς των δύο συναρτήσεων, ενδέχεται να προβληματιστήκατε για το πώς ακριβώς χρησιμοποιούνται και δεν αποκλείεται να σκεφτήκατε ότι είναι κάπως δύσχρηστες. Όπως είπαμε και πριν, όμως, αυτές οι συναρτήσεις καλούνται αυτόματα! Δείτε τις παρακάτω γραμμές και θα καταλάβετε αμέσως τι εννοούμε:

```
$aCar = new car() ;
$aCar->doors = 1 ;

echo $aCar->doors ;
```

Στην πρώτη γραμμή δημιουργούμε το αντικείμενο aCar και στη δεύτερη προσπαθούμε να τροποποιήσουμε τη μεταβλητή doors. Ε, λοιπόν, επειδή η μεταβλητή doors έχει οριστεί σαν protected, τη στιγμή που προσπαθούμε να αλλάξουμε την τιμή της καλείται στο παρασκήνιο η συνάρτηση __set(). Σαν πρώτη παράμετρος δίνεται αυτόματα το όνομα της μεταβλητής στην οποία προσπαθούμε να επέμβουμε (doors) και σαν δεύτερη παράμετρος δίνεται η τιμή που θέλουμε να καταχωρήσουμε στη μεταβλητή (1). Ξαναδείτε τώρα πώς έχουμε ορίσει τη συνάρτηση __set(). Επειδή

προσπαθούμε να δώσουμε στη μεταβλητή doors μια τιμή μικρότερη του 2, η συνάρτηση __set() θα εκχωρήσει τελικά την τιμή 2. Στην τρίτη από τις παραπάνω γραμμές προσπαθούμε να διαβάσουμε τη τιμή της doors. Κι επειδή πρόκειται για μεταβλητή που έχει οριστεί σαν protected, θα κληθεί σιωπηρά η συνάρτηση __get(). Σε αυτήν δίνεται ως παράμετρος το όνομα της μεταβλητής που προσπαθούμε να διαβάσουμε (doors). Ο ορισμός της __get() δεν επιβάλλει κανενός είδους φιλτράρισμα κι έτσι θα επιστρέψει την τιμή 2 (αυτή αποδόθηκε τελικά στη μεταβλητή doors).

Ανακύκλωση κώδικα

Μια από τις βασικότερες έννοιες του αντικειμενοστραφούς προγραμματισμού, ως προς την δυνατότητα επαναχρησιμοποίησης του κώδικα, είναι αυτή της **κληρονομικότητας** (inheritance). Η PHP υποστηρίζει την κληρονομικότητα με τη χρήση της λέξης κλειδιού **extends**, η οποία τοποθετείται αμέσως μετά το όνομα της νέας κλάσης. Μετά από τη λέξη extends ακολουθεί το όνομα της γονικής κλάσης, από την οποία κληρονομούνται όλες οι ιδιότητες και οι μέθοδοι που έχουν οριστεί σαν public ή protected:

```
class sportCar extends car{
     $color = "yellow"
     $doors = "2"
}
```

Παραπάνω φαίνεται ο ορισμός της κλάσης sportCar, σαν απόγονος (descendant) της κλάσης car[2]. Έτσι, αν και δεν ορίσαμε εκ νέου τη συνάρτηση changeColor() μέσα στην κλάση sportCar, θα κληρονομηθεί από την κλάση car και θα μπορούμε να τη χρησιμοποιήσουμε κανονικά. Παρατηρείστε κάτι ακόμα: Η μεταβλητή color είχε οριστεί σαν private εντός της car και δεν κληρονομείται από την sportCar. Επομένως, είναι αναμενόμενο να την ορίσουμε ξανά. Τι συμβαίνει όμως με τη doors; Η συγκεκριμένη μεταβλητή είχε οριστεί σαν protected μέσα στην κλάση car και άρα κληρονομήθηκε. Μήπως υπάρχει πρόβλημα που την ορίζουμε ξανά μέσα στον απόγονο της car; Όπως υποψιάζεστε, όχι. Η PHP επιτρέπει την **επικάλυψη** μεταβλητών και μεθόδων της γονικής κλάσης, από ομώνυμες μεταβλητές και μεθόδους του απογόνου. Έτσι, για τα αντικείμενα τύπου sportCar η μεταβλητή $doors ορίζεται εκ νέου και έχει εξ ορισμού την τιμή 2 (όχι την τιμή 5 που είχαμε ορίσει στη γονική κλάση car).

Η επικάλυψη αποτελεί δίκοπο μαχαίρι. Υπάρχουν περιπτώσεις που μας εξυπηρετεί, αλλά υπάρχουν και περιπτώσεις που μπορεί να οδηγήσει σε εσφαλμένες αλλοιώσεις στις τιμές μεταβλητών! Γι' αυτό το λόγο η PHP διαθέτει τη λέξη κλειδί **final**, με την οποία μπορούμε να απενεργοποιήσουμε τη δυνατότητα επικάλυψης. Η συγκεκριμένη λέξη κλειδί τοποθετείται μπροστά από τον ορισμό της μεταβλητής ή της μεθόδου που δεν θέλουμε να επικαλύπτεται από τις κλάσεις-απογόνους.

Εάν έρχεστε από τον κόσμο της C++ ενδέχεται να απογοητευτείτε (προσωρινά), καθώς η PHP δεν υποστηρίζει πολλαπλή κληρονομικότητα. Κάθε κλάση μπορεί να έχει μόνο μια γονική κλάση, από την οποία και κληρονομεί χαρακτηριστικά. Όσοι από εσάς έχετε γνώσεις Java, μπορείτε να φανταστείτε την απάντηση της PHP σε αυτό

2. Η κλάση car ονομάζεται και γονική κλάση (parent class) για την sportCar.

το θέμα. Η λύση δίνεται με τα λεγόμενα **interfaces** (διασυνδέσεις). Οι διασυνδέσεις είναι κάτι σαν "καλούπια" για τη δημιουργία κλάσεων. Μέσα σε ένα interface δηλώνουμε ένα σύνολο συναρτήσεων, οι οποίες *πρέπει* να υλοποιηθούν σε όλες τις κλάσεις που χρησιμοποιούν το συγκεκριμένο interface. Ας δούμε ένα παράδειγμα:

```
interface vehicle{
     public function run() ;
}
```

Μόλις ορίσαμε μια διασύνδεση με όνομα vehicle, χρησιμοποιώντας τη λέξη κλειδί **interface**. Στο σώμα της διασύνδεσης δηλώνουμε τη συνάρτηση run, χωρίς ωστόσο να την υλοποιούμε. Σημειώστε ότι οι συναρτήσεις που δηλώνουμε σε ένα interface πρέπει να είναι πάντοτε τύπου public! Τώρα, αν θέλουμε η κλάση car να αξιοποιεί αυτή τη διασύνδεση, πρέπει να χρησιμοποιήσουμε τη λέξη κλειδί **implements** κάπως έτσι:

```
Class car implements vehicle{
     function run(){
               …
               …
     }
     …
}
```

Σημειώστε ότι, εφόσον η κλάση car αξιοποιεί το interface με όνομα vehicle, πρέπει να περιλαμβάνει μια υλοποίηση για όλες τις συναρτήσεις που περιλαμβάνει το συγκεκριμένο interface[3]. Αναρωτιέστε τώρα σε τι εξυπηρετούν οι διασυνδέσεις; Παρόλο που μία κλάση στην PHP δεν μπορεί να έχει πολλές γονικές κλάσεις, μπορεί να χρησιμοποιεί πολλά interfaces! Έτσι, μπορούμε να ορίζουμε κλάσεις με πολλούς διαφορετικούς προγόνους.

Τελειώνοντας τη σύντομη περιπλάνηση στον κόσμο του αντικειμενοστραφούς προγραμματισμού σε PHP, αξίζει να σημειώσουμε κάτι τελευταίο. Θυμόσαστε πώς ορίζουμε τις σταθερές σε ένα πρόγραμμα της PHP; Στο προηγούμενο μέρος της σειράς είχαμε πει ότι κάτι τέτοιο γίνεται με τη βοήθεια της λέξης κλειδί define. Αυτό δεν ισχύει για τις κλάσεις! Για να ορίσουμε μια σταθερά μέσα σε μία κλάση, πρέπει να χρησιμοποιήσουμε τη λέξη κλειδί const. Ας δούμε ένα παράδειγμα:

```
class car{
     const wheels = 4 ;
     …
}
```

3. Στο παράδειγμά μας υπάρχει μόνο μια τέτοια συνάρτηση (η run).

Προσέξτε ότι, σε αντίθεση με τις μεταβλητές, δεν χρησιμοποιούμε το σύμβολο του δολαρίου μπροστά από το όνομα μιας σταθεράς. Επιπρόσθετα, για να αναφερθούμε σε μια σταθερά μιας κλάσης δεν χρησιμοποιούμε τον τελεστή βέλος (->) αλλά τη διπλή άνω-κάτω τελεία (::) Έτσι, για να διαβάσουμε την τιμή της σταθεράς wheels, πρέπει να δώσουμε κάτι τέτοιο:

```
$myVar = car::wheels ;
echo "A car have $ myVar wheels" ;
```

Συμπερίληψη κώδικα

Σε ότι αφορά την "ανακύκλωση" κώδικα μεταξύ ξεχωριστών προγραμμάτων, η PHP παρέχει δύο εναλλακτικές προτάσεις. Αυτές είναι οι **require('/some_path/file.php')** και **include('/some_path/file.php')**. Τα περιεχόμενα του αρχείου "file.php" μπορεί να είναι κώδικας σε PHP, σε HTML ή και σκέτο κείμενο! Οι require και include χρησιμοποιούνται συνήθως για να ενσωματώνουμε στα προγράμματα είτε βιβλιοθήκες είτε μεμονωμένα κομμάτια κώδικα. Για παράδειγμα, όταν στήνουμε ένα ολόκληρο site μπορούμε να δημιουργήσουμε ένα αρχείο με κώδικα PHP που θα υλοποιεί την επικεφαλίδα του site (header) κι ένα αντίστοιχο για το υποσέλιδο (footer). Στη συνέχεια, σε κάθε αρχείο PHP που υλοποιεί μια σελίδα του site, μπορούμε να περιλαμβάνουμε τα αρχεία για το header και για το footer, ώστε όλες οι σελίδες να έχουν την ίδια επικεφαλίδα και το ίδιο υποσέλιδο χωρίς να γράφουμε και να ξαναγράφουμε τα ίδια κομμάτια κώδικα. Με αυτόν τον τρόπο, αν αργότερα θελήσουμε να αλλάξουμε την επικεφαλίδα του site, θα αρκεί να επέμβουμε μόνο σε ένα αρχείο. Τέλος, η συμπεριφορά των require και include δεν είναι πανομοιότυπη. Η διαφορά τους εκδηλώνεται όταν προσπαθούμε να συμπεριλάβουμε κάποιο αρχείο που δεν υπάρχει. Στην περίπτωση χρήσης της require δημιουργείται σφάλμα και σταματά η εκτέλεση του κώδικα, ενώ η include εμφανίζει απλώς μια προειδοποίηση.

Δεδομένα από φόρμες

Αφού γνωρίσαμε τα κυριότερα δομικά συστατικά της PHP, είναι καιρός να δούμε τον συνηθέστερο μηχανισμό με τον οποίο λαμβάνουμε δεδομένα από το χρήστη. Ως επί το πλείστον, η επικοινωνία του χρήστη με μια εφαρμογή PHP γίνεται μέσα από φόρμες HTML. Ας εξετάσουμε μια απλή περίπτωση φόρμας HTML, που ζητά από το χρήστη να δηλώσει το όνομά του:

```
<html>
   <head>
      <title>Welcome form</title>
   </head>
   <body>
      <form name="userin" action="hello.php" method="post">
         <input type="text" name="usrname" size="10">
         <input type="submit">
```

```
        </form>
    </body>
</html>
```

Εδώ έχουμε μια φόρμα με το όνομα userin, η οποία στέλνει τα δεδομένα της στο αρχείο hello.php του web server με τη μέθοδο **post**. Η εν λόγω μέθοδος αποστέλλει τα δεδομένα μέσα στο header ενός αιτήματος HTTP. Όπως θα έχετε υπόψη σας υπάρχει και η μέθοδος **GET**, η οποία στέλνει τα δεδομένα ενσωματώνοντάς τα σε ένα κατάλληλα σχηματισμένο URL (πχ: http://url/code.php?field1_name =data1& field2_name =data2&…). Έτσι αποστέλλονται και τα δεδομένα στους servers της Google, κάθε φορά που πραγματοποιούμε μια αναζήτηση.

Όπως είπαμε ήδη, η φόρμα του παραδείγματος στέλνει τα δεδομένα της στη σελίδα hello.php. Ας δούμε τον κώδικα αυτής της σελίδας. Για λόγους οικονομίας χώρου, θα εστιάσουμε μόνο στο τμήμα του κώδικα που σχετίζεται με τη φόρμα και λαμβάνει κι επεξεργάζεται τα δεδομένα:

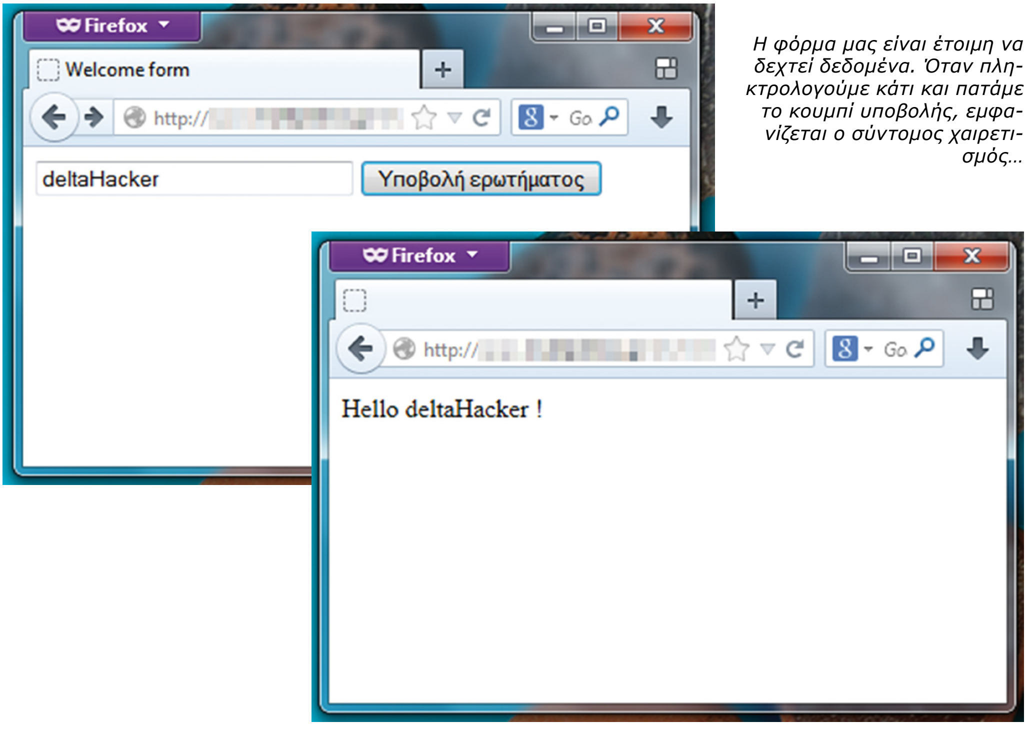
```
<?php
$name = $_POST['usrname'] ;
echo "Hello $name !" ;
?>
```

Η μεταβλητή name λαμβάνει την τιμή της από το στοιχείο "usrname", ενός μυστήριου πίνακα με όνομα _POST. Που βρέθηκε αυτός ο πίνακας; Όπως είπαμε, η μέθοδος post στέλνει τα δεδομένα (τα ονόματα των πεδίων και τις αντίστοιχες τιμές) μέσα στο header ενός αιτήματος HTTP. Ο μεταφραστής της PHP με τη σειρά του, λαμβάνει τα δεδομένα και τα αποθηκεύει σε έναν πίνακα με όνομα _POST. Σε αυτόν τον πίνακα, τα ονόματα των πεδίων που στάλθηκαν από τη φόρμα χρησιμοποιούνται σαν δείκτες για τις αντίστοιχες τιμές. Παρεμπιπτόντως, το ίδιο συμβαίνει και όταν αποστέλλονται δεδομένα με τη μέθοδο GET, με τη διαφορά ότι τότε σχηματίζεται ο πίνακας _GET. Σημειώστε ότι ανεξάρτητα από τη μέθοδο που χρησιμοποιείται κάθε φορά, σχηματίζεται πάντα και ο πίνακας _REQUEST. Σε αυτόν αποθηκεύονται όσα δεδομένα στάλθηκαν μέσα σε header καθώς και όσα στάλθηκαν μέσω του URL.

Όπως έχουμε ξαναπεί η PHP είναι ιδιαίτερα ελαστική — με ό,τι καλό ή κακό συνεπάγεται αυτό. Έτσι, για την πρόσβαση στα δεδομένα που στέλνονται με τη μέθοδο POST, παρέχονται δύο ακόμα μηχανισμοί: Ο ένας προβλέπει τη χρήση του πίνακα ονόματι HTTP_POST_VARS[`field_name’] και ο άλλος προβλέπει τη χρήση της μεταβλητής $field_name, που δημιουργείται αυτόματα για κάθε πεδίο της φόρμας. Εδώ πρέπει να πούμε ότι κανένας από αυτούς τους μηχανισμούς δεν είναι ενεργοποιημένος εξ ορισμού. Για να αξιοποιήσουμε κάποιον, πρέπει προηγουμένως να επέμβουμε στο αρχείο ρυθμίσεων του interpreter της PHP και να τον ενεργοποιήσουμε. Πάντως, η χρήση του πρώτου μηχανισμού περιλαμβάνει περισσότερη πληκτρολόγηση για το τίποτα, ενώ η χρήση του δεύτερου ενέχει σοβαρούς κινδύνους ασφάλειας. Αν ένας κακόβουλος χρήστης μάθει ή μαντέψει τα ονόματα των μεταβλητών που χρη-

σιμοποιούμε θα μπορεί να αλλάξει τις τιμές τους κατά βούληση, σχηματίζοντας τα κατάλληλα αιτήματα HTTP. Αντιλαμβάνεστε πόσο επικίνδυνο θα ήταν κάτι τέτοιο;

Για την πρόσβαση στα δεδομένα μις φόρμας προτείνουμε τον τρόπο που δείξαμε στο παράδειγμα (μέθοδος POST). Μάλιστα, οι τιμές που λαμβάνουμε θα ήταν φρόνιμο να διέρχονται πάντα από κάποια ρουτίνα που θα κάνει το λεγόμενο input sanitization. Βλέπετε, οτιδήποτε λαμβάνουμε από τον χρήστη πρέπει να αντιμετωπίζεται ως πιθανά επικίνδυνο!



Η φόρμα μας είναι έτοιμη να δεχτεί δεδομένα. Όταν πληκτρολογούμε κάτι και πατάμε το κουμπί υποβολής, εμφανίζεται ο σύντομος χαιρετισμός...

Φιλτράρισμα εισόδου

Η PHP προσφέρει αρκετές συναρτήσεις που μπορούμε να χρησιμοποιήσουμε για το φιλτράρισμα των δεδομένων. Οι πιο απλές αφαιρούν ή "απενεργοποιούν" τους χαρακτήρες και τα string που έχουν ειδική ερμηνεία για την HTML. Μερικές τέτοιες συναρτήσεις είναι η strip_tags, η htmlspecialchars και η htmlentities. Ας δούμε τι προσφέρει η καθεμία:

Η **strip_tags($string, [$allowable_tags])** αφαιρεί από την παράμετρο $string τους χαρακτήρες null καθώς και τις ετικέτες HTML και PHP. Στην προαιρετική παράμετρο $allowable_tags μπορούμε να καθορίσουμε τις ετικέτες HTML που θέλουμε να επιτρέπονται.

Η **htmlspecialchars ($string)** κωδικοποιεί τους ειδικούς χαρακτήρες σε μορφή HTML entities. Οι "οντότητες HTML" αποτελούν μια περιγραφική μορφή των χαρακτήρων. Με τη βοήθειά τους μπορούμε να περιγράφουμε χαρακτήρες, οι οποίοι χάνουν την όποια ειδική ερμηνεία επιδέχονται και απλώς εμφανίζονται. Σκεφτείτε

για παράδειγμα το χαρακτήρα "<", με τον οποίο ξεκινούν οι ετικέτες της HTML. Ο συγκεκριμένος χαρακτήρας δεν μπορεί να τοποθετηθεί οπουδήποτε σε ένα έγγραφο HTML, καθώς ο browser θα προσπαθήσει να μεταφράσει το κείμενο που ακολουθεί σαν ετικέτα. Για να εμφανίσουμε το συγκεκριμένο χαρακτήρα χωρίς να προκύψει ζήτημα με τις ετικέτες, πρέπει να χρησιμοποιήσουμε το αντίστοιχο HTML entity (<). Η συνάρτηση htmlspecialchars πραγματοποιεί ακριβώς αυτές τις μετατροπές. Εντοπίζει τους ειδικούς χαρακτήρες (&, ", ', < και >) και τους αντικαθιστά με τα αντίστοιχα HTML entities. Για την πλήρη τεκμηρίωση της συνάρτησης μπορείτε να ανατρέξετε εδώ `www.php.net/manual/en/function.htmlspecialchars.php`.

Παρόμοια σε λειτουργία είναι και η **htmlentities($string)**. Αυτή η συνάρτηση, όμως, δεν επενεργεί μόνο τους ειδικούς χαρακτήρες. Αντικαθιστά όλους τους χαρακτήρες για τους οποίους υπάρχει κάποιο HTML entity.

Εκτός από αυτές τις συναρτήσεις, η PHP διαθέτει πολλές ακόμα, οι οποίες είναι περισσότερο εξειδικευμένες και αποσκοπούν στο input validation και sanitization. Μια απ' αυτές είναι η **filter_var (mixed $variable [, int $filter = FILTER_DEFAULT [, mixed $options]])**. Η εν λόγω συνάρτηση δέχεται τα προς έλεγχο δεδομένα στη μεταβλητή $variable και εφαρμόζει σ' αυτά το φίλτρο που προσδιορίζει η μεταβλητή $filter. Φυσικά, οι επιτρεπόμενες τιμές για τη μεταβλητή $filter είναι συγκεκριμένες, όπως και τα φίλτρα που υλοποιεί η συνάρτηση. Μπορείτε να μάθετε για τα φίλτρα που δέχεται αυτή η ενδιαφέρουσα συνάρτηση από το `www.php.net/manual/en/filter.filters.php`.

Κανονικό φιλτράρισμα

Μια ακόμα μέθοδος φιλτραρίσματος των δεδομένων είναι αυτή με τη βοήθεια των regular expressions (κανονικές εκφράσεις). Πώς είπατε; Δεν έχετε ασχοληθεί ποτέ στα σοβαρά με τις κανονικές εκφράσεις; Να μια καλή ευκαιρία να μελετήσετε τα άρθρα περί κανονικών εκφράσεων (τεύχη 021 έως και 024). Η PHP υποστηρίζει πλήρως τόσο τη διάλεκτο ERE (POSIX Extended) όσο και την PCRE (Perl-Compatible). Για αναζητήσεις μοτίβων με τη διάλεκτο ERE προσφέρονται οι συναρτήσεις ereg (string $pattern , string $string) και eregi (string $pattern , string $string). Η μόνη διαφορά τους είναι ότι η δεύτερη δεν κάνει διάκριση μεταξύ πεζών και κεφαλαίων (είναι case insensitive). Για τη διάλεκτο PCRE προσφέρεται η συνάρτηση preg_match (string $pattern , string $string). Βέβαια, για την εργασία που μας ενδιαφέρει (το φιλτράρισμα) χρειαζόμαστε συναρτήσεις που θα υποστηρίζουν τις κανονικές εκφράσεις αλλά δεν θα περιορίζονται σε μια απλή αναζήτηση. Συγκεκριμένα, χρειαζόμαστε συναρτήσεις που θα πραγματοποιούν αντικαταστάσεις. Τέτοιες είναι οι ereg_replace, eregi_replace και preg_replace (δείτε, π.χ., το `www.php.net/manual/en/function.preg-replace.php`). Τέλος, αν και μπορούμε να χρησιμοποιούμε οποιαδήποτε διάλεκτο των κανονικών εκφράσεων θέλουμε, η ομάδα ανάπτυξης της PHP συνιστά την PCRE.

Ολοκληρώνοντας κι αυτό το άρθρο, το ταξίδι μας στον κόσμο της PHP έχει προχωρήσει αρκετά. Η καταιγίδα των πληροφοριών που δεχθήκαμε ενδέχεται να ήταν λίγο κουραστική και να μας ζάλισε, αλλά τώρα τα πανιά μας φούσκωσαν και η κατασκευή της πρώτης μας εφαρμογής WEB αχνοφαίνεται στον ορίζοντα! Υπομονή μέχρι το επόμενο τεύχος, οπότε θα συνεχίσουμε το ταξίδι μας. Ως τότε, λίγη εξάσκηση με την αντικειμενοστραφή πλευρά της PHP θα μας φέρει ακόμα πιο κοντά στη θολή γραμμή του ορίζοντα...

Skill: Beginner
Tags: transparent proxy, anonymous proxy, distorting proxy, elite proxy, SSH tunnels, Tor, anonymity



Χωρίς (τη δική μας) IP!

Εντάξει, η επικοινωνία χωρίς IP σε ένα δίκτυο χτισμένο στο TCP/IP είναι απλά αδύνατη. Μπορούμε όμως να κρύψουμε τη διεύθυνση IP του υπολογιστή μας. Μπορούμε να βάλουμε κάποιον άλλον "μπροστά" και σε κάθε μας κίνηση να φαίνεται αυτός. Αν αυτός που θα αναλάβει τον συγκεκριμένο ρόλο είναι έμπιστος κι εχέμυθος, η ιδιωτικότητά μας θα παραμένει προστατευμένη...

by Thiseas

Πολύ συχνά δεχόμαστε το εξής ερώτημα: Πόσο κοντά μου μπορεί να φτάσει κάποιος που γνωρίζει τη διεύθυνση IP του μηχανήματός μου; Η αναζήτηση με βάση τη διεύθυνση IP μπορεί να οδηγήσει κάποιον μέχρι την εταιρεία παροχής Internet (ISP, Internet Service Provider) που χρησιμοποιούμε. Από εκεί και ύστερα, το αν αυτός που ψάχνει θα φτάσει κυριολεκτικά έξω από την πόρτα μας εξαρτάται από το αν η εταιρεία (ISP) θα του διαθέσει τα στοιχεία μας. Στην Ελλάδα, οι εταιρείες του χώρου δίνουν πολύ δύσκολα τα στοιχεία των χρηστών τους. Συγκεκριμένα, για να κάνουν κάτι τέτοιο απαιτούν εισαγγελική εντολή! Επομένως, σύμφωνα με τη νομοθεσία της χώρας μας, μόνο οι διωκτικές αρχές θα μπορούσαν να εντοπίσουν το σπίτι μας. Δυστυχώς, όμως, έχουμε δει και περιπτώσεις στις οποίες αυτός που ψάχνει έχει έναν "κολλητό" στον ISP που τυγχάνει να χρησιμοποιούμε κι έτσι καταφέρνει να μάθει τα πάντα. Ας αφήσουμε όμως αυτή τη σπάνια περίπτωση κατά μέρος κι ας δούμε τι μπορεί να μάθει για εμάς, κάποιος που γνωρίζει μόνο τη διεύθυνση IP:

- Τη χώρα μας
- Πιθανά την πόλη μας
- Τα στοιχεία του ISP μας

Ο ίδιος "κάποιος", αν έχουμε συνδεθεί σε αυτόν ή αν με κάποιον τρόπο βλέπει τα πακέτα που στέλνουμε, μπορεί να μάθει και τα ακόλουθα:

- Το λειτουργικό σύστημα ή/και τον web browser που χρησιμοποιούμε.
- Από ποιο site τον επισκεφτήκαμε (π.χ., από ένα της Google).

Νομικό πλαίσιο

Στο άρθρο 5 του Νόμου 2472/1997 (προσωπικά δεδομένα) προβλέπεται ότι η χορήγηση των στοιχείων σε τρίτους, χωρίς να το γνωρίζει ο συνδρομητής, εναπόκειται στην κρίση του ISP. Δηλαδή, ο ISP έχει την διακριτική ευχέρεια να ενημερώσει ή όχι τον συνδρομητή για την αποκάλυψη των στοιχείων του σε τρίτους. Λόγο έχει και η Αρχή Προστασίας Προσωπικών Δεδομένων (http://www.dpa.gr). Στο άρθρο 11 του τροποποιημένου (ίδιου) νόμου 2472/1997, προβλέπεται ότι ο χρήστης (συνδρομητής) πρέπει να ενημερώνεται για την επικείμενη ανακοίνωση των προσωπικών του στοιχείων, πριν ενημερωθούν οι τρίτοι! Μπορεί όμως να γίνει άρση αυτής της υποχρέωσης για ενημέρωση, όταν πρόκειται **για θέματα εθνικής ασφαλείας ή για την εξακρίβωση ιδιαίτερα σοβαρών εγκλημάτων** (http://bit.ly/whatthelawsays).

Η αποστολή μας

Κατά την άποψη του συντάκτη, τα μοντέρνα λειτουργικά συστήματα θα έπρεπε να δίνουν τη δυνατότητα προστασίας όλων των προσωπικών δεδομένων του χρήστη. Δυστυχώς όμως δεν το κάνουν. Έχετε αναρωτηθεί το γιατί; Μήπως δεν μπορούν; Μήπως δεν συμφέρει κάποιους; Για να μην μπούμε σε μονοπάτια σεναρίων συνωμοσίας και κατασκοπίας, τα οποία καθόλου δεν απέχουν από την πραγματικότητα, καλύτερα να εστιάσουμε στο αντικείμενο μας. Α!, παραλίγο να το ξεχάσω: Hello NSA! We love you ;)

Όπως καταλαβαίνετε, σε αυτό το άρθρο θα προσπαθήσουμε να πάρουμε την κατάσταση στα χέρια μας. Θα δούμε με ποιον τρόπο μπορούμε να διασφαλίσουμε την ιδιωτικότητά μας. Πριν συνεχίσουμε, όμως, θα κάνουμε τις εξής δύο παραδοχές: Καταρχάς, το άρθρο γράφτηκε με στόχο την υπεράσπιση των βασικών ελευθεριών του χρήστη και όχι την απόκρυψη της ταυτότητας εγκληματιών, τρομοκρατών κι άλλων αρρωστημένων ομάδων. Άλλωστε και η μεθοδολογία που θα ακολουθήσουμε έχει προσαρμοστεί σε αυτό ακριβώς το πλαίσιο. Θεωρούμε επίσης ότι ο χρήστης δουλεύει σε Windows, αν και αυτά που θα δούμε είναι ανεξάρτητα από το λειτουργικό σύστημα.

Υπάρχουν αρκετοί τρόποι για να κρύψουμε τη διεύθυνση IP του μηχανήματός μας. Θα αναφέρουμε τους πιο διαδεδομένους, που θεωρούμε ότι είναι και οι πιο αποδοτικοί. Όλοι αυτοί οι τρόποι στηρίζονται στην ίδια γενική ιδέα: Θα "εμφανιζόμαστε" στο Internet με τη διαμεσολάβηση κάποιου άλλου υπολογιστή και, τέλος πάντων, σαν να ήμασταν αυτός ο κάποιος άλλος. Ο υπολογιστής που κάνει τη διαμεσολάβηση ονομάζεται proxy server ή, πιο απλά, σκέτο proxy. Όσοι δικτυακοί τόποι έχουν την περιέργεια να μάθουν ποιοι είμαστε κι από πού ερχόμαστε, θα καταλήγουν στον proxy και κάπου εκεί η έρευνά τους θα προσκρούει (ιδανικά) σε τοίχο. Σκεφτείτε τώρα ο proxy-server που χρησιμοποιούμε να βρίσκεται κάπου στο Ανατολικό Τιμόρ ;)



Εικόνα 1
Ένα web application που επιτρέπει να χρησιμοποιήσουμε κάποιον proxy server, χωρίς να αλλάξουμε κάποια ρύθμιση στο σύστημα.



Εικόνα 2
Δοκιμάζουμε το web application του HideMyAss με το προγραμματάκι μας. Τα αποτελέσματα είναι ικανοποιητικά...

Κατηγορίες μεσολαβητών

Ας μιλήσουμε λίγο για τους proxy servers. Όταν χρησιμοποιούμε έναν proxy για να επικοινωνήσουμε με τον έξω κόσμο, είναι σαν να λέμε το εξής: Αγαπητέ κύριε Proxy X, μπορείς σε παρακαλώ να με πας στο Google; (Ή οπουδήποτε αλλού.) Ο proxy εκτελεί αμέσως την εντολή μας και στέλνει στο Google ένα σχετικό HTTP request. Το Google, τουλάχιστον τις περισσότερες φορές, απαντά κανονικά και ο proxy προωθεί σε εμάς αυτή την απάντηση. Έτσι, καταφέρνουμε να επικοινωνήσουμε με το Google, χωρίς να μιλήσουμε άμεσα μαζί του. Εδώ όμως βρίσκεται μια σημαντική λεπτομέρεια: Το HTTP request που στέλνει ο proxy ενδέχεται να περιλαμβάνει κρίσιμες πληροφορίες για μας, που δεν θέλουμε να γνωστοποιηθούν. Γενικά, κάθε proxy χαρακτηρίζεται από έναν τύπο, ανάλογα με το πόσες πληροφορίες αφήνει να διαρρεύσουν μέσω των HTTP requests. Βλέπετε, τα αιτήματα HTTP περιέχουν τις λεγόμενες μεταβλητές HTTP, ορισμένες από τις οποίες μεταφέρουν κρίσιμες πληροφορίες:

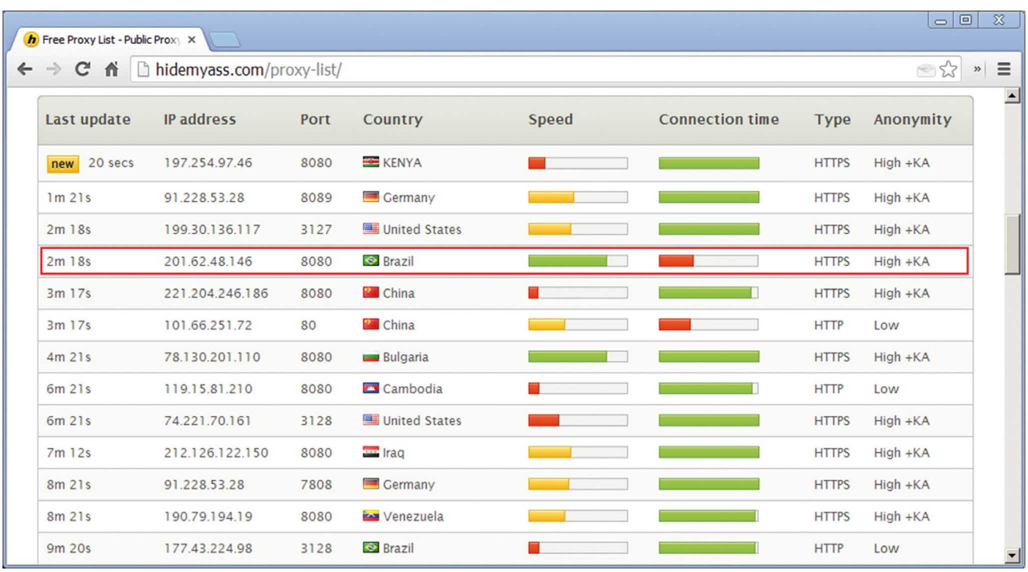
- **REMOTE_ADDR**. Είναι η IP address αυτού που κάνει την κλήση (εφόσον μεσολαβεί proxy πρόκειται για τη δική του IP κι όχι για τη δική μας).

- **HTTP_VIA**. Όταν είναι ορισμένη αυτή η μεταβλητή, φέρει τη διεύθυνση IP του proxy. Επομένως, ο ορισμός της μαρτυρά ότι όντως χρησιμοποιείται κάποιος proxy.
- **HTTP_X_FORWARDED_FOR**. Όταν είναι ορισμένη αυτή η μεταβλητή, περιέχει τη δική μας διεύθυνση IP!
- **HTTP_ACCEPT_LANGUAGE**. Η γλώσσα που χρησιμοποιείται στον web browser από τον οποίο στάλθηκε το HTTP request.
- **HTTP_USER_AGENT**. Ο web browser (όνομα και έκδοση) από τον οποίο στάλθηκε το HTTP request.
- **HTTP_HOST**. Το όνομα του μηχανήματος από το οποίο ξεκίνησε το HTTP request.

Όπως καταλαβαίνετε, δεν είναι όλοι οι proxies εχέμυθοι, ούτε κατάλληλοι για να προστατεύσουν την ιδιωτικότητά μας. Ανάλογα με το πόσες πληροφορίες δίνουν για εμάς, χωρίζονται στις ακόλουθες κατηγορίες:

- **Transparent Proxy**. Πρόκειται για τους proxies που δεν κρύβουν καμία πληροφορία και ταυτόχρονα ενημερώνουν το συνομιλητή μας (δηλαδή τον απομακρυσμένο server) για τη διαμεσολάβησή τους.
- **Anonymous Proxy**. Πρόκειται για τους proxies που δεν φανερώνουν τη δική μας διεύθυνση IP κι άρα σχηματίζουν αιτήματα HTTP με την μεταβλητή **HTTP_X_FORWARDED_FOR** χωρίς τιμή.

Όπως αντιλαμβάνεστε, το ενδιαφέρον επικεντρώνεται στη δεύτερη κατηγορία. Όμως, ο ορισμός που δώσαμε για αυτή την κατηγορία είναι αρκετά γενικός. Μιλήσαμε για την τιμή μίας μόνο μεταβλητής και δεν είπαμε τίποτα για τις άλλες. Οι anonymous proxies, λοιπόν, μπορούν να χωριστούν σε επιμέρους κατηγορίες:



Last update	IP address	Port	Country	Speed	Connection time	Type	Anonymity
new 20 secs	197.254.97.46	8080	KENYA			HTTPS	High +KA
1m 21s	91.228.53.28	8089	Germany			HTTPS	High +KA
2m 18s	199.30.136.117	3127	United States			HTTPS	High +KA
2m 18s	201.62.48.146	8080	Brazil			HTTPS	High +KA
3m 17s	221.204.246.186	8080	China			HTTPS	High +KA
3m 17s	101.66.251.72	80	China			HTTP	Low
4m 21s	78.130.201.110	8080	Bulgaria			HTTPS	High +KA
6m 21s	119.15.81.210	8080	Cambodia			HTTP	Low
6m 21s	74.221.70.161	3128	United States			HTTPS	High +KA
7m 12s	212.126.122.150	8080	Iraq			HTTPS	High +KA
8m 21s	91.228.53.28	7808	Germany			HTTPS	High +KA
8m 21s	190.79.194.19	8080	Venezuela			HTTPS	High +KA
9m 20s	177.43.224.98	3128	Brazil			HTTP	Low

Εικόνα 3
HideMyAss: O server που φιλοξενεί μια ολόκληρη λίστα από free proxies!

- **Simple anonymous proxies**. Αυτοί οι proxies κρύβουν τη δική μας διεύθυνση IP και στη θέση της δίνουν τη δική τους. Κατά τα άλλα, δεν κρύβουν το ρόλο τους (ως διαμεσολαβητές).

  ```
  REMOTE_ADDR = proxy IP
  HTTP_VIA = proxy IP
  HTTP_X_FORWARDED_FOR = proxy IP
  ```

- **Distorting proxies**. Οι proxies αυτοί κρύβουν τη δική μας διεύθυνση IP και στη θέση της δίνουν μια τυχαία! Κατά τα άλλα, ούτε κι αυτοί κρύβουν το ρόλο τους.

  ```
  REMOTE_ADDR = proxy IP
  HTTP_VIA = proxy IP
  HTTP_X_FORWARDED_FOR = κάποια άσχετη τυχαία IP
  ```

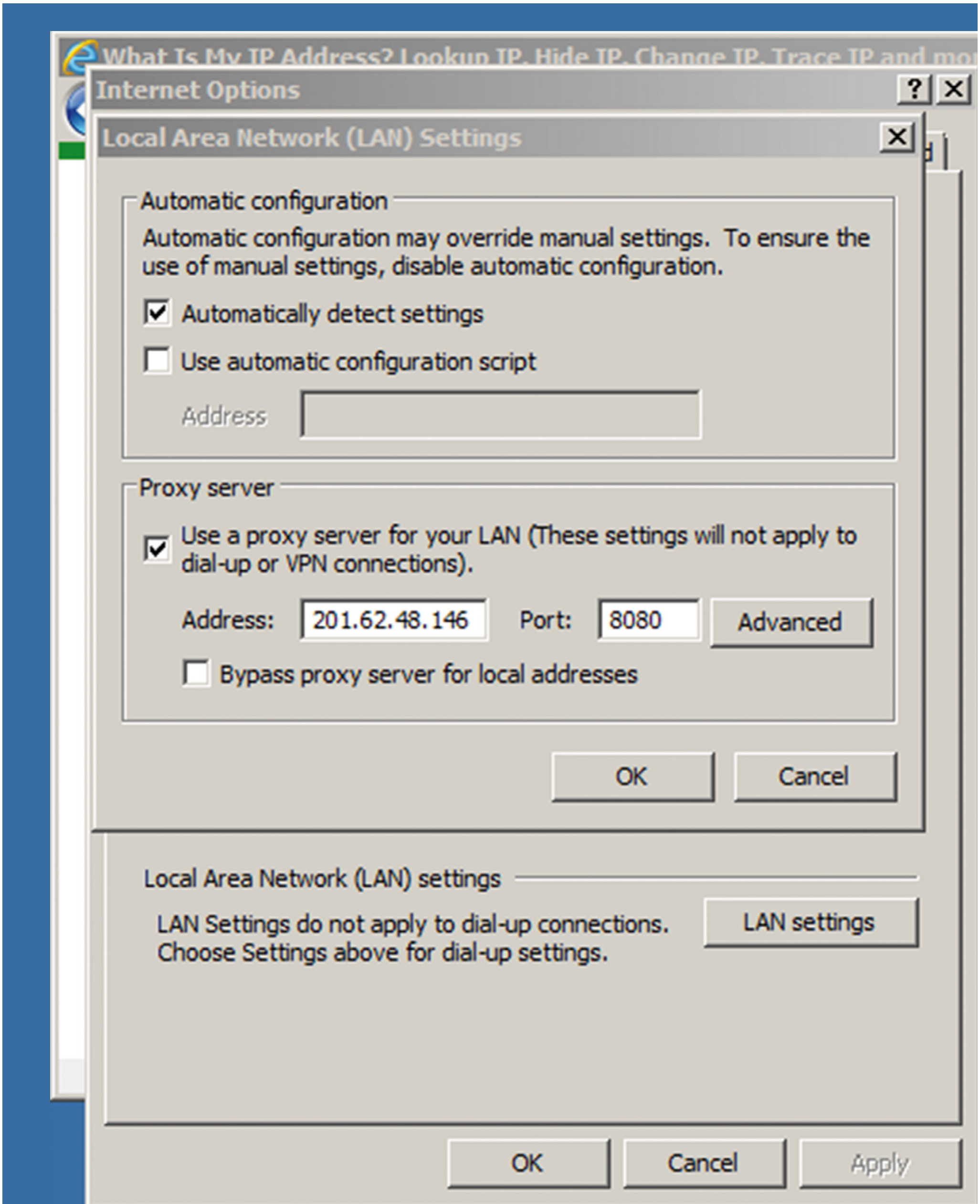
- **High anonymity proxies (ή elite proxies)**. Οι proxies της κατηγορίας προσφέρουν τη μέγιστη δυνατή "προστασία". Κρύβουν τόσο τη δική μας διεύθυνση όσο και τον ρόλο τους, αφήνοντας τις αντίστοιχες μεταβλητές χωρίς τιμή.

  ```
  REMOTE_ADDR = proxy IP
  HTTP_VIA = τίποτα
  HTTP_X_FORWARDED_FOR = τίποτα
  ```

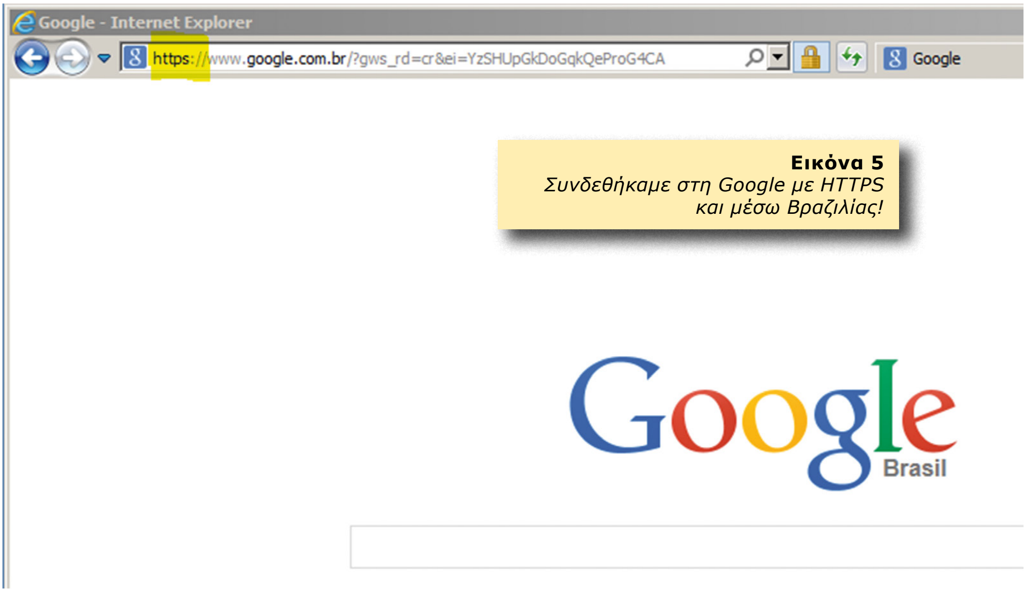
Πολύ σημαντική για την προστασία των προσωπικών μας δεδομένων είναι και η κρυπτογραφημένη επικοινωνία (http://en.wikipedia.org/wiki/Secure_Sockets_Layer). Στην περίπτωση του web, αυτό προϋποθέτει σύνδεση μέσω **HTTPS** και όχι μέσω **HTTP**. Σε μια κρυπτογραφημένη επικοινωνία δεν μπορεί να παρεισφρήσει κάποιος ανάμεσα σε εμάς και στον δικτυακό συνομιλητή μας και, κατ' επέκταση, δεν μπορεί να υποκλέψει τα δεδομένα που ανταλλάσσουμε. (Σ.τ.Ε. Υπάρχουν βέβαια κι εξαιρέσεις: http://deltahacker.gr/?p=8973) Έτσι, αν θέλουμε τη μέγιστη δυνατή προστασία, πρέπει να επιλέξουμε κάποιον **elite proxy** και να πραγματοποιούμε συνδέσεις μέσω **HTTPS**.

Περί αξιοπιστίας

Όπως πάντα, στην πράξη τα πράγματα είναι πολύ πιο περίπλοκα απ' όσο ακούγονται. Αρκετά sites διακόπτουν την επικοινωνία όταν αντιλαμβάνονται την ύπαρξη κάποιου proxy. Αυτό δεν συμβαίνει από ιδιοτροπία, ούτε εξαιτίας της γενικής παραξενιάς του ιδιοκτήτη του εκάστοτε site. Το μπλοκάρισμα των proxies αποτελεί μια άμυνα των servers απέναντι στα ανόητα script kiddies, που στο παρελθόν εξαπέλυαν ατελείωτες επιθέσεις με προκάλυμμα κάποιον proxy. Επίσης, πολλά sites δεν υποστηρίζουν καθόλου την κρυπτογραφημένη σύνδεση HTTPS. Τέλος, κανένας δεν μπορεί να μας εγγυηθεί ότι ο τάδε proxy είναι απόλυτα αξιόπιστος κι ότι δεν προσφέρει τις πληροφορίες μας σε όποιον πληρώνει. Επομένως, θα ήταν προτιμότερο να χρησιμοποιούμε proxies που προσφέρονται από γνωστές, έμπιστες εταιρείες. Αυτοί οι proxies μπορούν να θεωρηθούν περισσότερο αξιόπιστοι από έναν proxy server στην Κολομβία, που προσφέρει κάποιος άγνωστος Χόρχε Κριμινάλ ;) Για παράδειγμα, το hidemyass αποτελεί μια αρκετά καλή περίπτωση. Βέβαια, ούτε και τότε θα έχουμε εξασφαλίσει την ανωνυμία μας κατά 100%. Αν κάποια διωκτική Αρχή απαιτήσει δεδομένα από τους servers μιας εταιρείας, να είσαστε βέβαιοι ότι θα τα πάρει. Για την ώρα, όμως,



Εικόνα 4
Από εδώ γίνεται η ρύθμιση του proxy για τον Internet Explorer. Η ίδια ρύθμιση ισχύει και για τις περισσότερες δικτυακές εφαρμογές των Windows, ανάμεσα στις οποίες ανήκει και ο Chrome.



Εικόνα 5
Συνδεθήκαμε στη Google με HTTPS και μέσω Βραζιλίας!

οι διωκτικές Αρχές δεν ενδιαφέρονται για τις καταναλωτικές μας συνήθειες. Εκτός κι αν τρώτε ανθρώπους, δηλαδή. Επομένως, στο βαθμό που δεν κάνουμε κάτι παράνομο, με έναν elite proxy από γνωστή εταιρία και με μια κρυπτογραφημένη σύνδεση, θα είμαστε απόλυτα καλυμμένοι!

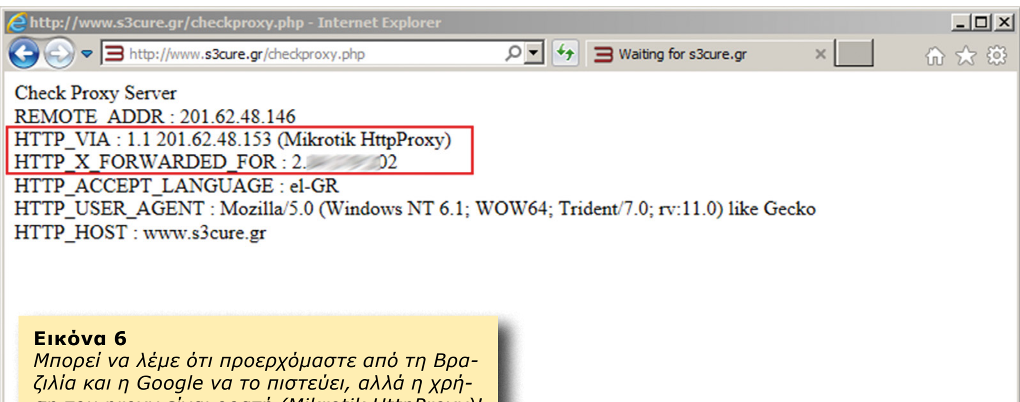
Πριν προχωρήσουμε, πρέπει ν' απαντήσουμε σε ένα κρίσιμο ερώτημα: Πώς θα διαπιστώσουμε ότι ο proxy που χρησιμοποιούμε κάνει όλα όσα υπόσχεται; Ο πιο σίγουρος τρόπος είναι να φτιάξουμε μια απλή εφαρμογή σε PHP, η οποία θα μας πληροφορεί για το περιεχόμενο των μεταβλητών που αναφέραμε νωρίτερα. Βέβαια, αυτή η εφαρμογή θα πρέπει να τρέχει σε κάποιο VPS ή, τέλος πάντων, σε κάποιο μηχάνημα στο οποίο θα συνδεθούμε μέσω του proxy! Το προγραμματάκι για αυτή τη δουλειά θα μπορούσε να είναι κάπως έτσι:

```
<?php
   echo 'Check Proxy Server';
   echo '<br /> REMOTE_ADDR          : '.$_SERVER['REMOTE_ADDR'];
   echo '<br /> HTTP_VIA             : '.$_SERVER['HTTP_VIA'];
   echo '<br /> HTTP_X_FORWARDED_FOR : '.$_SERVER['HTTP_X_FORWARDED_FOR'];
   echo '<br /> HTTP_ACCEPT_LANGUAGE : '.$_SERVER['HTTP_ACCEPT_LANGUAGE'];
   echo '<br /> HTTP_USER_AGENT      : '.$_SERVER['HTTP_USER_AGENT'];
   echo '<br /> HTTP_HOST            : '.$_SERVER['HTTP_HOST'];
?>
```

Φυσικά, εκτός απ' αυτή τη λύση υπάρχει και η ετοιματζίδικη: Μπορούμε να επισκεφτούμε ορισμένα sites, τα οποία είναι κατασκευασμένα ειδικά για αυτούς τους ελέγχους. Τέτοια sites είναι τα ακόλουθα:

- http://www.whatismyip.com
- http://whatismyipaddress.com
- http://checkip.dyndns.org
- http://whatip.com

Ξέρουμε τι σκέφτεστε τώρα: Και ποιος μας λέει ότι αυτά τα sites δεν παρέχουν παραπλανητικές πληροφορίες, με σκοπό να καλύψουν τις ψευδείς υποσχέσεις των proxy servers; Αν και αυτό το σενάριο αποτελεί μια απίθανη θεωρία συνωμοσίας, κανένας δεν μπορεί να εγγυηθεί το αντίθετο! Η εναλλακτική, λοιπόν, είναι στήσουμε τη δική μας εφαρμογή στο δικό μας server. Τώρα, έχοντας κατά νου όλα όσα αναφέραμε μέχρι τώρα, μπορούμε να δούμε τέσσερις τρόπους για την προστασία ανωνυμίας ή/και προσωπικών δεδομένων.



Εικόνα 6
Μπορεί να λέμε ότι προερχόμαστε από τη Βραζιλία και η Google να το πιστεύει, αλλά η χρήση του proxy είναι ορατή (Mikrotik HttpProxy)!

Τρόπος 1: Web proxy server

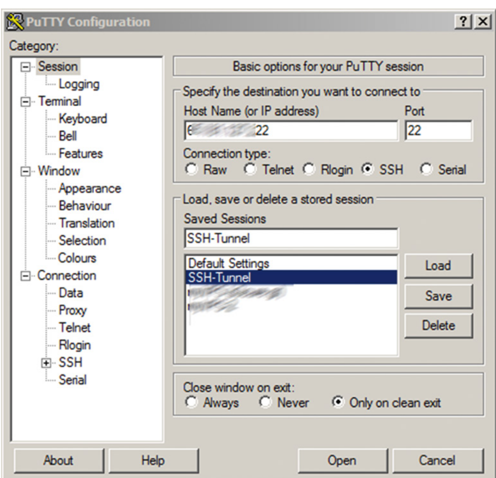
Ο πιο απλός και γρήγορος τρόπος για να καλύψουμε το ποιοι είμαστε και από πού ερχόμαστε, είναι με τη βοήθεια ορισμένων proxies που παρέχουν τις υπηρεσίες τους μέσα από μία εφαρμογή web. Το μόνο που έχουμε να κάνουμε είναι να επισκεφτούμε το αντίστοιχο site και να δώσουμε στο κατάλληλο text box το URL που θέλουμε να επισκεφτούμε. Τα υπόλοιπα γίνονται αυτόματα (εικόνα 1). Στο παράδειγμά μας χρησιμοποιήσαμε τον web proxy της HideMyAss, ο οποίος είναι τύπου Elite ενώ υποστηρίζει και συνδέσεις HTTPS. Ο ίδιος server παρέχει κι άλλες ευκολίες: Επιλογή του τρόπου απόκρυψης του server, "κρυπτογράφηση" ή κωδικοποίηση του URL, απενεργοποίηση της JavaScript κ.ά. Εμείς κάναμε κι έναν μικρό έλεγχο αξιοπιστίας, με το προγραμματάκι σε PHP που παρουσιάσαμε προηγουμένως. Τα αποτελέσματα ήταν θετικότατα! Ο proxy δεν μαρτυράει την πραγματική μας διεύθυνση IP, εκπέμποντας τις μεταβλητές HTTP_VIA και HTTP_X_FORWARDED_FOR χωρίς τιμή. Επιπρόσθετα, σαν διεύθυνση δηλώνει την 67.159.5.242, η οποία σύμφωνα με το whois (http://whois.domaintools.com/67.159.5.242) βρίσκεται κάπου στο Σικάγο των ΗΠΑ. Καθόλου κακό.

Τρόπος 2: Free Proxy Servers

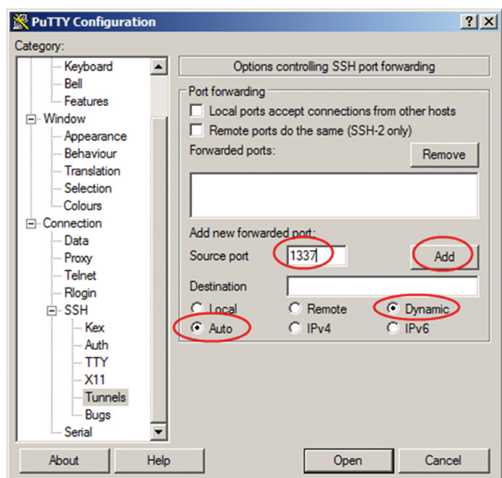
Εκτός από τους web proxies υπάρχουν κι εκείνοι που μπορούμε να χρησιμοποιούμε πιο μόνιμα, με μια μικρή αλλαγή στις ρυθμίσεις του web browser. Υπάρχουν πολλοί τέτοιοι proxy servers και οι περισσότεροι είναι δωρεάν (εικόνα 3). Γενικά, οι απαιτούμενες ρυθμίσεις στον browser είναι αρκετά απλές. Αυτό που χρειάζεται να γνωρίζουμε είναι η IP του proxy server και η δικτυακή θύρα στην οποία δέχεται συνδέσεις. Ας δούμε τη διαδικασία αλλαγής στους δημοφιλέστερους browsers:

- **Internet Explorer**. Στο μενού ακολουθούμε τη διαδρομή Tools → Internet Options → Connections → LAN Settings. Στο παράθυρο που ανοίγει τσεκάρουμε το Use proxy server for your LAN... και δίνουμε στο πεδίο Address την IP του server και στο πεδίο Port την αντίστοιχη δικτυακή θύρα (εικόνα 3). Με αυτή τη ρύθμιση στον Internet Explorer, ο proxy θα χρησιμοποιείται από όλες σχεδόν τις δικτυακές εφαρμογές του συστήματος, μεταξύ των οποίων είναι κι ο Chrome.

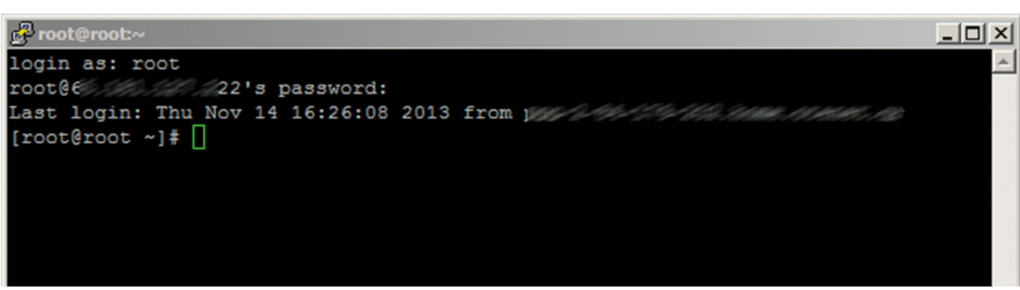
- **Chrome**. Στο μενού ακολουθούμε τη διαδρομή Settings → Show Advanced Settings → Change Proxy Settings → LAN Settings. Με αυτόν τον τρόπο καταλήγουμε στο παράθυρο που εμφανίζεται και μέσα από τον Internet Explorer.



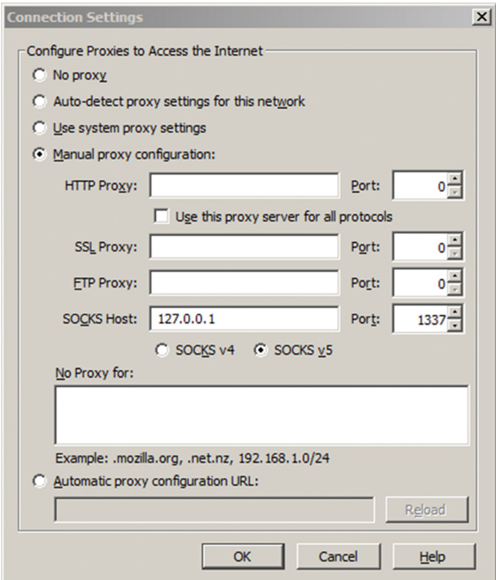
Εικόνα 7
Δηλώνουμε στο PuTTY τη διεύθυνση του server μας...



Εικόνα 8
Ορίζουμε τις απαραίτητες παραμέτρους...



Εικόνα 9
Πατάμε στο Open και μπαίνουμε στον απομακρυσμένο SSH server (εμείς συνδεθήκαμε ως root)...

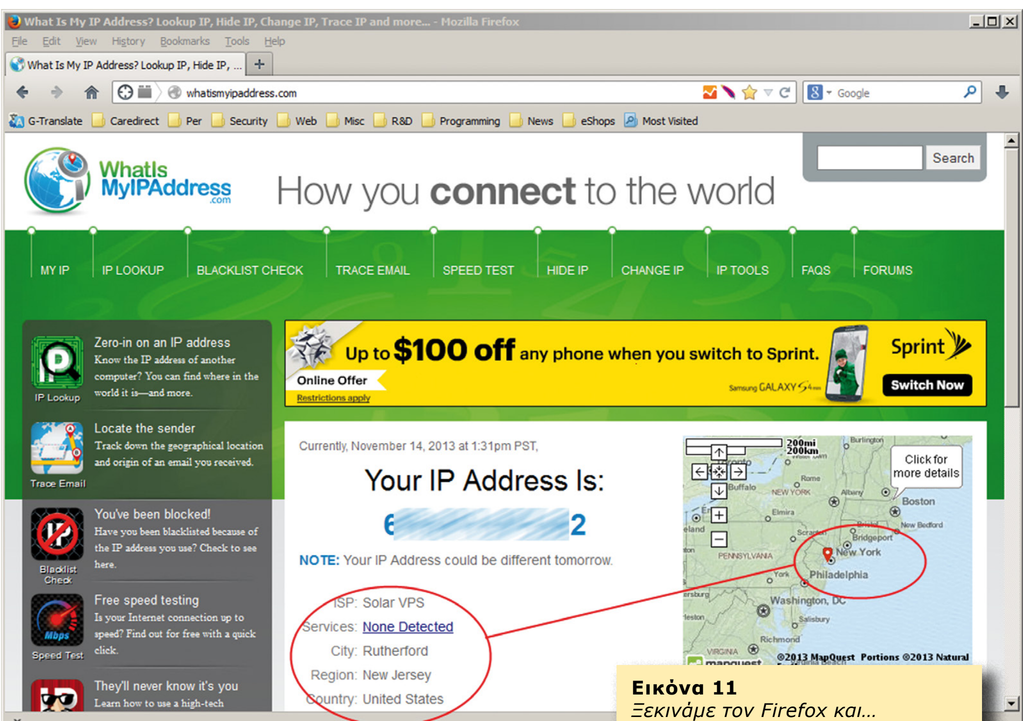


Εικόνα 10
Ορίζουμε στα settings του Firefox τη διεύθυνση IP και το port του server μας...

Επομένως, τσεκάρουμε το Use proxy server for your LAN... και συμπληρώνουμε με τον ίδιο τρόπο. Σημειώστε ότι η ρύθμιση θα ισχύσει και για τον Internet Explorer.

- **Firefox**. Στο μενού ακολουθούμε τη διαδρομή Tools → Options → Advanced → Network → Settings. Επιλέγουμε το Manual proxy configuration και βάζουμε στο πεδίο HTTP proxy τη διεύθυνση IP του proxy και στο πεδίο Port την αντίστοιχη δικτυακή θύρα. Τέλος τσεκάρουμε την επιλογή Use this proxy server for all protocols και πατάμε OK.

Να σημειώσουμε εδώ ότι κατά τις δοκιμές μας χρησιμοποιήσαμε έναν proxy server από τη Βραζιλία (εικόνα 5). Με μία επίσκεψη στο δικό μας πρόγραμμα, όμως, διαπιστώσαμε ότι κάτι δεν πήγαινε



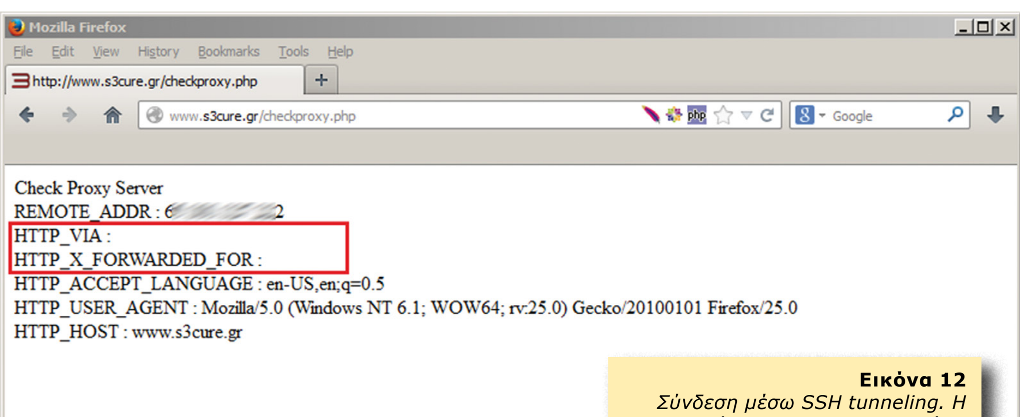
Εικόνα 11
Ξεκινάμε τον Firefox και... "Welcome from New Jersey"!

καθόλου καλά (εικόνα 6). Ο server παραείναι ομιλητικός για τα γούστα μας! Ο Βραζιλιάνικος server φανερώνει το γεγονός ότι λειτουργεί ως μεσολαβητής, ενώ φανερώνει και την πραγματική μας διεύθυνση IP. Όλα αυτά έρχονται σε πλήρη αντίθεση με τον ισχυρισμό ότι αποτελεί High Anonymity Proxy. Προσοχή λοιπόν!

Τρόπος 3: SSH Tunneling

Ο τελευταίος τρόπος είναι λίγο πιο... freak oriented ;) Θα κάνουμε χρήση μιας σύνδεσης SSH προς κάποιο δικό μας VPS. Μάλιστα, θα εκμεταλλευτούμε μια λειτουργία που παρέχει ο πολύ καλός και δωρεάν SSH client ονόματι PuTTY (http://bit.ly/getPuTTY). Αυτή η μέθοδος έχει το καλό ότι είναι κρυπτογραφημένη από την αρχή ως τη... μέση — μέχρι τον VPS, δηλαδή. Ο server μας όμως θα λειτουργεί σαν proxy κι από 'κει και πέρα θα εμφανίζεται αυτός αντί για εμάς! Για να πετύχουμε όλα αυτά, ο VPS μας θα πρέπει να τρέχει Linux και να έχουμε πρόσβαση στη γραμμή εντολών μέσω SSH. Μια αναλυτική παρουσίαση για τη σύνδεση μέσω SSH Tunneling έχει γίνει και εδώ http://deltahacker.gr/?p=4773. Ωστόσο, οι εικόνες 8 έως 12 δείχνουν τα βασικά βήματα σε περιβάλλον Windows. Προσοχή χρειάζεται στην εικόνα 8 και συγκεκριμένα εκεί όπου ορίζουμε τη δικτυακή θύρα (port) ως dynamic. Αν αποθηκεύσουμε το session με την χρήση του Save που παρέχει το PuTTY και αργότερα το φορτώσουμε με το Load (εικόνα 7), πρέπει να σετάρουμε ξανά την πόρτα ως dynamic! Το PuTTY μοιάζει να έχει κάποιο bug και δεν αποθηκεύει το port σαν dynamic αλλά ως local...

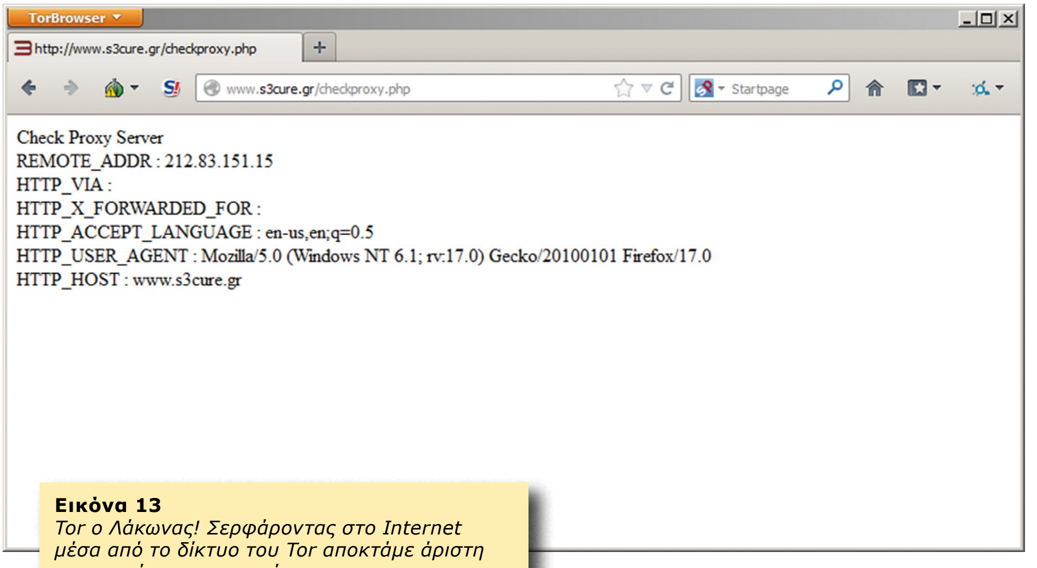
Αφού στήσαμε το δικό μας SSH tunnel, κάναμε και πάλι μια δοκιμή με τη δική μας εφαρμογή. Τα αποτελέσματα ήταν πολύ καλά: Η πραγματική μας διεύθυνση IP δεν φαινόταν πουθενά (εικόνα 12). Το SSH tunneling αποτελεί τον κλασικό τρόπο με τον οποίο μπορεί κανείς να παρακάμψει ένα εταιρικό firewall και να βγει στο Internet ανενόχλητος! (Σ.τ.Ε. Με το SSH tunneling κρυπτογραφούμε τη δικτυακή κίνηση συγκεκριμένων εφαρμογών, όπως, π.χ., είναι ο web browser. Αν επιθυμούμε την κρυπτογράφηση της δικτυακής κίνησης ενός ή περισσοτέρων υπολογιστών, τότε μπορούμε να στραφούμε στο OpenVPN. Δείτε τρία διαφορετικά άρθρα, για τρεις διαφορετικές περιπτώσεις χρήσης: http://deltahacker.gr/?p=5187, http://deltahacker.gr/?p=5669 και http://deltahacker.gr/?p=9608)



Εικόνα 12
Σύνδεση μέσω SSH tunneling. Η μεταβλητή REMOTE_ADDR περιέχει τη διεύθυνση του proxy server μας.

Τρόπος 4: Μέσω Tor

Πρόκειται για την αγαπημένη μέθοδο προστασίας των απανταχού χάκερ αλλά κι όλων των "σκοτεινών" τύπων του darknet. Το Tor αποτελεί πλέον ένα ολοκληρωμένο πακέτο και είναι πανεύκολο να το σετάρει κανείς στα Windows (www.torproject.org/docs/tor-doc-windows.html.en). Επιγραμματικά μπορούμε να πούμε ότι πρόκειται για ένα δίκτυο από πάρα πολλούς proxies. Τα πακέτα μας διέρχονται από μια αλυσίδα τέτοιων proxies, η οποία σχηματίζεται κάθε φορά τυχαία. Μάλιστα, κάθε proxy γνωρίζει μόνο τα στοιχεία του προηγούμενου και του επόμενου. Έτσι, η εύρεση της πραγματικής μας διεύθυνσης IP καθίσταται πρακτικά αδύνατη! Η δοκιμή με τη μικρή εφαρμογή μας δεν έδειξε κάτι αντίθετο (εικόνα 13). Δυστυχώς, το Tor έχει και ένα σημαντικό μειονέκτημα: Σέρνεται απερίγραπτα! (Σ.τ.Ε. Μπορούμε να περνάμε όλη ανεξαιρέτως τη δικτυακή κίνηση TCP ενός ολόκληρου τοπικού δικτύου μέσα από το Tor: http://deltahacker.gr/?p=9405)



Εικόνα 13
Tor ο Λάκωνας! Σερφάροντας στο Internet μέσα από το δίκτυο του Tor αποκτάμε άριστη προστασία της ανωνυμίας μας.

Συμπεράσματα

Παρουσιάσαμε τέσσερις βασικούς τρόπους με τους οποίους μπορείτε να κρύψετε την IP σας. Υπάρχουν κι άλλοι. Δεν θα πρέπει να ξεχάσουμε, π.χ., τη χρήση VPN, μέσω του οποίου ο proxy server γίνεται ένα ακόμα node στο LAN του σπιτιού σας ή της δουλειά σας. Φυσικά, να πούμε ότι μπαίνοντας από IPs άλλων χωρών αναγκαστικά εντασσόμαστε στους νόμους της εκάστοτε χώρας, δηλαδή τόσο στις υποχρεώσεις όσο και στα δικαιώματα. Οπότε μην σας φανεί περίεργο αν ο browser αρχίσει να σας αφήνει να μπαίνετε σε sites που πριν δεν ήταν δυνατόν να μπείτε. Τέλος, δεν θα σας πούμε ποιος είναι ο καλύτερος τρόπος από τους παραπάνω, διότι απλά αυτό εξαρτάται από το τι ακριβώς θέλετε να κάνετε. Όπως πάντα, από ένα σημείο και μετά οι αποφάσεις είναι σίγουρα δικές σας.